



**Universidade do Estado do Rio de Janeiro**

Centro de Tecnologia e Ciências

Instituto de Matemática e Estatística

Yuri Bastos Gabrich

**A blockchain application to pave the way for transactive energy  
at Brazilian micro/minи-grids**

Rio de Janeiro

2019

Yuri Bastos Gabrich

**Uma aplicação blockchain para proporcionar a comercialização de energia em  
micro/minigerações distribuídas brasileiras**

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Igor Machado Coelho  
Coorientador: Dr. Vitor Nazário Coelho

Rio de Janeiro  
2019

## CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC-A

G117

Gabrich, Yuri Bastos.

A blockchain application to pave the way for transactive energy at  
Brazilian micro/mini-grids/ Yuri Bastos Gabrich. - 2019.

183f. : il.

Orientador: Igor Machado Coelho.

Coorientador: Vítor Nazário Coelho.

Dissertação (Mestrado em Ciências Computacionais) - Universidade  
do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Energia elétrica – Comercialização – Teses. 2. Geração  
distribuída de energia elétrica – Teses. 3. Software – Controle de  
qualidade. I. Coelho, Igor Machado. II. Coelho, Vítor Nazário. III.  
Universidade do Estado do Rio de Janeiro. Instituto de Matemática e  
Estatística. IV. Título.

CDU 620.9

Patricia Bello Meijinhos CRB7/5217 -Bibliotecária responsável pela elaboração da ficha catalográfica

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta  
dissertação, desde que citada a fonte

---

Assinatura

---

Data

Yuri Bastos Gabrich

**Uma aplicação blockchain para proporcionar a comercialização de energia em  
micro/minigerações distribuídas brasileiras**

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Aprovada em 19 de outubro de 2019.

Banca Examinadora:

---

Prof. Dr. Igor Machado Coelho (Orientador)  
Instituto de Matemática e Estatística – UERJ

---

Dr. Vitor Nazário Coelho (Coorientador)  
OptBlocks Ltda

---

Prof.<sup>a</sup> Dra. Maria Clicia Stelling de Castro  
Instituto de Matemática e Estatística – UERJ

---

Prof. Dr. Diego Nunes Brandão  
CEFET/RJ

Rio de Janeiro  
2019

## ACKNOWLEDGEMENTS

First of all, I'd like to thank all of those who have been waiting for the end of this tough course. The supports came on different matters from each particular effort someone can make. Fortunately, patience was key for all of us to conduct learning over each other mistakes.

I would like to thank both advisors, Igor Machado Coelho and Vitor Nazário Coelho, for the time given to develop the skills for self-criticism, writing, and workarounds out of the barriers faced on the way. Special thanks also go to their life partners, Cristiane Tavares and Thays Aparecida de Oliveira, for random conversations on unusual meetings.

Moreover, this thesis would have been impossible without the contributions of Luis (Nando) Ochoa, Luiz Alberto Fortunato, Sérgio Mafra, and Nivalde José de Castro. With the sharing of their scarce time to appoint me valuable reports and to allow discussions about the subject at the beginning of the present work, they had contributed to reaching further horizons. Similar appreciation goes for Alexandre Sztajnberg, Luciano Porto Barreto, Maria Clicia Stelling de Castro, and Diego Nunes Brandão, who have integrated both examination committees, the mid-term and the final one, and have given important guidance to the research structure and content presentation.

Also, I am particularly grateful for the support given by professors and assistants from the Computational Science department that were willing to help me during the whole pathway. Likewise, I would like to express my very great appreciation to colleagues for sharing their knowledge far beyond what classrooms may offer.

I can't forget to mention my dear friends from IEEE too. They have experienced the challenges to finish the master's thesis from the beginning of the study until the last reviewed line. Struggle times persevered with the events we have either organized or attended together. A big thank you to all volunteers that I met and worked with.

Above all, I wish to acknowledge the help provided by my closest relatives and friends. At every uncertain day, they kept strong the support to reach the desired outcome. The support that is only given by few but that surpasses in commitment and confidence the difficulties imposed by others.

New technologies never come alone. It's a package: technological changes, followed by social, political and cultural changes.

*Alvin Toffler (*The Third Wave*)*

## RESUMO

GABRICH, Yuri Bastos *Uma aplicação blockchain para proporcionar a comercialização de energia em micro/minigerações distribuídas brasileiras.* 2019. 183 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

O consumo de energia elétrica está diretamente relacionado à qualidade de vida e bem-estar da população. Portanto, esforços gerais vêm acontecendo para suprir a crescente demanda por energia por meio de alternativas mais sustentáveis e socialmente responsáveis. Desde grandes usinas centralizadas à Geração Distribuída (GD) de eletricidade, o avanço da Tecnologia da Informação e Comunicação (TIC) está trazendo novas abordagens para o setor elétrico, apesar dos desafios técnicos e comerciais para se fazer isso. Mais recentemente, o advento da tecnologia blockchain ressurgiu um desejo antigo de livre comércio de eletricidade entre todos os tipos de consumidores de energia. Especificamente, no nível de distribuição, o conceito de *Energia Transativa* (TE) é crucial para qualquer discussão sobre técnicas para se lidar com um novo subconjunto de mercado de energia. Assim, o presente trabalho é uma quase-POC que delinea os desafios e oportunidades de uma aplicação distribuída (DApp) para apoiar o gerenciamento de energia de micro/mini-redes dentro de uma determinada legislação brasileira, que denomina o grupo de consumo compartilhado. Esta dissertação também produziu o contrato inteligente MTEsm que gerencia a cripto-moeda SEB. Todo o processo de desenvolvimento é apresentado e tenta mostrar as questões econômicas e técnicas que devem ser consideradas para uma proposição completa de POC.

Palavras-chave: Blockchain. Comercialização de energia. Geração distribuída de energia. Micro/minigeração distribuída.

## ABSTRACT

GABRICH, Yuri Bastos *A blockchain application to pave the way for transactive energy at Brazilian micro/minigrids.* 2019. 183 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

The electricity consumption is directly related to the population quality of life and well-being. Therefore, general efforts have been happening to supply the growing power demand through more sustainable and socially responsible alternatives. From centralized big power plants to Distributed Generation (DG) of electricity, the advance of Information and Communications Technology (ICT) is bringing new approaches to the electricity sector, despite the technical and business challenges to do so. More recently, the advent of the blockchain technology has resurfaced a past desire of free energy trading between all kinds of power consumers. Specifically, in the distribution level, the concept of Transactive Energy (TE) is imperative for any discussion about the techniques to deal with a new subset of an energy market. Hence, the present work is a quasi-POC that outlines the challenges and opportunities of a distributed application (DApp) to support the power management of micro/minigrids inside a particular Brazilian regulation framework, denominated as shared consumption group. This dissertation has also produced the smart contract MTEsm that manages the crypto-currency SEB. All the development process is presented and attempts to show the economic and technical subjects that must be considered for a complete POC proposition.

Keywords: Blockchain. Transactive energy. Distributed energy. Micro/minigrid.

## LIST OF FIGURES

Figure 1 - Classification of Distributed Generation (DG) under the Brazilian legislation, and its positioning in the power sector chain. . . . .	19
Figure 2 - The layers of an ordinary blockchain. . . . .	33
Figure 3 - The operational layers of a shareable generation group supported by a distributed application (DApp). . . . .	40
Figure 4 - Concepts for the development of a system. . . . .	41
Figure 5 - The structure of the Microgrid Transactive Energy smart contract (MTEsm) code. . . . .	57
Figure 6 - The flowchart of the MTEsm main interface. . . . .	62
Figure 7 - The pathway to be accepted in a group's private environment. . . . .	63
Figure 8 - The process to get information from the group (to be continued). . . . .	64
Figure 9 - The process to get information from the group (conclusion). . . . .	65
Figure 10 - A general voting procedure called by a member. . . . .	66
Figure 11 - The process of a new power plant crowdfunding. . . . .	67
Figure 12 - The process to update some information (to be continued). . . . .	68
Figure 13 - The process to update some information (continuity). . . . .	69
Figure 14 - The process to update some information (continuity). . . . .	70
Figure 15 - The process to update some information (conclusion). . . . .	71
Figure 16 - The process to increase the group power capacity (to be continued). . . . .	72
Figure 17 - The process to increase the group power capacity (continuity). . . . .	73
Figure 18 - The process to increase the group power capacity (conclusion). . . . .	74
Figure 19 - The trade agreement process between members. . . . .	75
Figure 20 - $M_3$ sells quotas to $M_1$ . . . . .	77
Figure 21 - The slight difference between the electricity sector concepts. . . . .	105
Figure 22 - The Unified Modeling Language (UML) class diagram of the DApp proposed. . . . .	115
Figure 23 - The code folding and cross-referencing of the MTEsm. . . . .	119
Figure 24 - The process to conduct the example. . . . .	151
Figure 25 - Essential steps before MTEsm deployment. . . . .	152
Figure 26 - The configuration of the MTEsm deployment. . . . .	153
Figure 27 - The identification of the outputs of the deploy. . . . .	155
Figure 28 - The RPC output of the deploy. . . . .	156
Figure 29 - The deploy transaction block. . . . .	157
Figure 30 - The transactions details of the MTEsm deploy in the ledger. . . . .	158
Figure 31 - Admission request of $M_1$ (to be continued). . . . .	159
Figure 32 - Admission request of $M_1$ (conclusion). . . . .	160

Figure 33 - Admission request of $M3$ .	161
Figure 34 - Vote of $M1$ in the referendum about the admission request of $M3$ .	162
Figure 35 - Admission result for the $M3$ request.	163
Figure 36 - The request to change $M1$ 's quota (to be continued).	165
Figure 37 - The request to change $M1$ 's quota (conclusion).	166
Figure 38 - $M1$ vote in the referendum about the change request of $M1$ .	167
Figure 39 - $M3$ vote in the referendum about the change request of $M1$ .	168
Figure 40 - The result for the $M1$ change request.	169
Figure 41 - The request to change $M3$ 's quota (to be continued).	170
Figure 42 - The request to change $M3$ 's quota (conclusion).	171
Figure 43 - $M3$ vote in the referendum about the change request of $M3$ .	172
Figure 44 - $M1$ vote in the referendum about the change request of $M3$ .	173
Figure 45 - The result for the $M3$ change request.	174
Figure 46 - The request of $M3$ to change $M1$ 's tokens (to be continued).	175
Figure 47 - The request of $M3$ to change $M1$ 's tokens (conclusion).	176
Figure 48 - $M1$ vote in the referendum about the change request for $M1$ .	177
Figure 49 - $M3$ vote in the referendum about the change request for $M1$ .	178
Figure 50 - The result for the $M3$ change request about $M1$ .	179
Figure 51 - The final values of $M1$ (to be continued).	180
Figure 52 - The final values of $M1$ (conclusion).	181
Figure 53 - The final values of $M3$ .	182

## LIST OF TABLES

Table	1 - Example of the DG management for a given time-frame for both group classifications. . . . .	21
Table	2 - Systems design decisions regarding (de)centralisation. . . . .	31
Table	3 - Blockchain applications in the electricity sector chain. . . . .	34
Table	4 - Comparison of some blockchains. . . . .	54
Table	5 - The differences between the 4 kinds of IDs handled by the MTEsm. . .	58
Table	6 - List of some identifications used along the example. . . . .	79
Table	7 - Financial analysis of the MTEsm. . . . .	82
Table	8 - The description of the constraints identified in the MTEsm (to be continued). . . . .	116
Table	9 - The description of the constraints identified in the MTEsm (continuity).117	
Table	10 - The description of the constraints identified in the MTEsm (conclusion).118	

## LIST OF ABBREVIATIONS AND ACRONYMS

**ACL** Free Contracting Environment

**ACR** Regulated Contracting Environment

**ANEEL** Brazilian Electricity Regulatory Agency

**API** Application Programming Interface

**BaaS** Blockchain as a Service

**CCEE** Electric Power Trading Chamber

**CMSE** Power Sector Monitoring Committee

**DApp** distributed application

**DBFT** Delegated Byzantine Fault Tolerance

**DER** Distributed Energy Resources

**DG** Distributed Generation

**DLT** Distributed Ledger Technology

**EPE** Energy Research Company

**EVM** Ethereum Virtual Machine

**ICO** Initial Coin Offering

**ICT** Information and Communications Technology

**IoT** Internet of Things

**MME** Brazilian Ministry of Mines and Energy

**MTEsm** Microgrid Transactive Energy smart contract

**NeoVM** Neo Virtual Machine

**ONS** National Power System Operator

**P2P** peer-to-peer

**PBFT** Practical Byzantine Fault Tolerance

**PoA** Proof of Authority

**POC** proof of concept

**PoET** Proof of Elapsed Time

**PoS** Proof of Stake

**PoW** Proof of Work

**SBFT** Simplified Byzantine Fault Tolerance

**SEB** Sharing Electricity in Brazil (crypto-currency)

**SEB** Brazilian Electricity Sector

**SIN** National Power Grid System

**TE** Transactive Energy

**TEE** Trusted Execution Environment

**UML** Unified Modeling Language

**UTC** Coordinated Universal Time

## CONTENTS

	<b>INTRODUCTION . . . . .</b>	13
1	<b>THE BRAZILIAN DISTRIBUTED GENERATION . . . . .</b>	18
2	<b>GENERAL CONCEPTS ABOUT BLOCKCHAIN APPLICATIONS . . . . .</b>	22
2.1	<b>Terminology and characteristics . . . . .</b>	23
2.1.1	<u>Basic properties . . . . .</u>	28
2.2	<b>Timeline . . . . .</b>	29
2.3	<b>The blockchain's varieties . . . . .</b>	29
2.3.1	<u>A framework of reference . . . . .</u>	32
2.4	<b>The distributed applications in the electricity sector . . . . .</b>	33
2.4.1	<u>New electricity market examples . . . . .</u>	35
3	<b>THE PROPOSED DISTRIBUTED APPLICATION . . . . .</b>	38
3.1	<b>Identification of the application requirements . . . . .</b>	41
3.2	<b>Choosing the right platform . . . . .</b>	43
3.2.1	<u>Bitcoin . . . . .</u>	43
3.2.2	<u>Ethereum . . . . .</u>	45
3.2.3	<u>Hyperledger . . . . .</u>	47
3.2.4	<u>Neo . . . . .</u>	48
3.2.5	<u>Corda . . . . .</u>	50
3.2.6	<u>Libra . . . . .</u>	51
3.2.7	<u>Comparison between the blockchains . . . . .</u>	52
3.3	<b>The DApp specifications . . . . .</b>	55
3.3.1	<u>The description of the operations . . . . .</u>	59
3.4	<b>An example of use . . . . .</b>	76
3.5	<b>Limitations and constraints in the development . . . . .</b>	78
3.6	<b>The group market size . . . . .</b>	80
3.7	<b>The costs of the smart contract's operations . . . . .</b>	81
	<b>CONCLUSION . . . . .</b>	85
	<b>REFERENCES . . . . .</b>	96
	<b>Appendix A – An overview of the electricity sector . . . . .</b>	105
	<b>Appendix B – The terms used throughout the DApp development . . . . .</b>	111
	<b>Appendix C – The UML class diagram . . . . .</b>	113
	<b>Appendix D – The descriptions of the considered constraints . . . . .</b>	116
	<b>Appendix E – The MTEsm . . . . .</b>	119
	<b>Appendix F – The setup of the TE example . . . . .</b>	151

## INTRODUCTION

The present work aims to depict how a novel technology, the blockchain, can substantially overcome the electricity sector challenges to integrate end-point consumers as active agents on the energy market. This section starts presenting the power energy access importance for social development and the hindrances identified on the literature to allow the integration of consumers of electricity as producers in the energy market. Then, brief arguments about blockchain showcase the available opportunity to solve the gap in the energy sector addressing the Brazilian micro/minи-grid context. Finally, the purpose, the methodology and the structure of this document are described.

### **The electricity relevance for society**

The worldwide growing demand for electricity is not solely a consequence of the population growth but by the rising of their well-being and quality of life (UN, 2017; IEA, 2017). Heavily influenced so far by the technological development, the energy base required to sustain these electricity demands is, economic and environmentally, unsustainable. Since the ending of the industrial era great advent, developed and developing nations are experiencing, on different paces, the switch from non-renewable, highly concentrated and centralized energy units with non-diversified power sources to alternative methods to handle with this energy issue (TOFFLER, 1980).

More than ever, a variety of solutions is being experienced at every part of the world. Until recently, most of them keeps the conservative infrastructure and market models, just replacing the power source fuels by renewable ones, as the case with solar and wind energy for example. They continue to be big power plants built up either on- and offshore, depending on each country land space and energy reservoir potential. Independently of the solution type, the challenge to assure electricity for each individual at a reasonable cost, respecting personal conditions, and being sustainable friendly is no small task.

However, the technological development also brings another special feature, the products price and size shrinking. This gives room to further discussions about the current business models towards market alternatives that had relied only on the imagination of few's. For instance, the Distributed Generation (DG) of electricity has been increasingly implemented to supply local power demands due to accessible costs of Distributed Energy Resources (DER) (IEA, 2017). Since big power plants are out of the scope to support the before-mentioned demands either because of infrastructure cost or environmental concerns (EPE, 2014a), it is expected a more active participation of end power consumers into the

energy base generation, which may influence on the economy and in the structure of the electricity sector (ELA et al., 2017; ZHENG et al., 2018; COELHO et al., 2017; CASTRO et al., 2018).

### A different option of energy access

The presence of DG also brings to the medium/low voltage grid the offer of electricity from prosumers (*producers + consumers*) beyond the traditional energy provided by the local power utility. A new electricity market design to support all this upcoming changes are being referenced as Transactive Energy (TE) in which novel ways to produce, buy, and sell electricity may happen through automated control (OLKEN, 2016). Indeed it is a system approach to facilitate the integration of an amount of DERs, to provide transparent energy prices, and to allow power consumers of all sizes to trade energy (FORFIA; KNIGHT; MELTON, 2016). For instance, end customers can now choose to trade with different power sources based on generation type, sustainable factors, and anything else over the average cost-basis of the electricity price.

A concept that remembers the sector restructuring discussions from the unbundling of vertically integrated power utilities to open the electricity market until low-voltage consumers but impracticable due to the technology of that time (SILVA, 2007; LITVINOV; ZHAO; ZHENG, 2019). Nowadays, this purpose arises from another perspective, with much more caution and business model options, besides the one-way solution of centralized control and operation of the power system.

Bearing this in mind, there is a continuous general effort for social integration into the renewable energy segment by means of the micro/mini-grid (JIAYI; CHUANWEN; RONG, 2008). A small scale and resilient electric grid connects the DGs with end consumers throughout the existing distribution grid, independently of the governance used to rule this relationship. There is a belief that this model “will bring a dynamic clean energy economy that empowers communities and customers – across all income levels, geographies, and demographics – to take control of their energy use, driving local economic growth and revitalization, improving the resiliency of our energy system, and protecting our environment” (MASIELLO, 2016, p.24). In other countries and locations there have also been initiatives for creating the so-called nanogrids, in which an optimized peer-to-peer topology (MAGNASCO et al., 2016) can be obtained by specialized real time management devices. In this sense, we could expect an ever more active TE.

Since 2010’s decade, several reports have been conducted to state challenges and plannings to implement such power generations. For instance, some of them are the *Technology Roadmap: Smart Grids* (IEA, 2011), the *Insertion of Distributed Photovoltaic Generation in Brazil: Conditioners and Impacts* (EPE, 2014b), and the *Energy Access*

*Outlook 2017: from poverty to prosperity* (IEA, 2017).

At first, electric technical issues used to be the claims against a distributed management and operation of the grid because it could create undesirable constraints on the electricity quality and reliability (LEW et al., 2017; ACKERMANN et al., 2017), and therefore impact on the electricity price definition and so forth (GIANELLONI et al., 2017). Nonetheless, the discussions mainly bump on different points of views about how market and business models should operate.

It has been previously observed that the current top-down business model is not appropriate to allow local market incentives to benefit its users in a transparent manner (HOMMELBERG et al., 2008), independently of the possible impacts on the grid. By contrast, other ones have already drawn attention to arguments against a market design rethink if the purpose is to maintain a central management and operation (HOGAN, 2019).

Apart from the discussion about who should operate the emerging DG and how it may impact the overall electricity price, the power system is becoming smarter with the constantly integration of Information and Communications Technologies (ICTs), the so-called smart grid (FARHANGI, 2009). However, Hommelberg et al. (2008) had identified four fundamental issues to implement ICT in power systems, which concerns with (i) *system architecture* to support the coordination between algorithm and physical control mechanism; (ii) *scalability*; (iii) *operational planning* to keep energy demand flexibility due to the huge intermittent power sources under consideration; and (iv) *timing discrepancy* of the communication process between planning and real-time operation.

## The blockchain importance to this subject

In this context, the blockchain has the basic requirements to integrate several power customer profiles into an unique informational network system able to exchange data about power generation, consumption and transactions without compromise the core business roles of the power utilities and customers, i.e., keeping the responsibility of the former to manage the electricity distribution and providing the best electricity service for the latter. By the way, blockchain applications can be developed to comply with local jurisdiction, even with DG technical specifications being almost the same worldwide.

The blockchain technology has been considered a revolution in the way the Internet is used. Although this view reminds the same feeling at the time the Internet itself began to become popular (FILIPPO; SZTAJNBERG, 1996), the blockchain remains on already established technologies such as peer-to-peer (P2P) network architecture and cryptographic keys to model a different Internet application paradigm (MOUGAYAR; BUTERIN, 2016). An innovation in the type and in the form that communication occurs

among its users, in which the transmitted data represents a certain information, but not the information content itself. And ensures its veracity and accessibility by a protocol that controls the data redundancy throughout a verification process done by specific users of the network.

Thereby, the blockchain technology may represent a new view for current management processes like the Internet had made several decades ago. Its applications have the potential to overcome saturated business models and to promote new ones, such as the case with the electricity sector, which has been experiencing an expansion of types, sizes and units of power generation.

Considering all of the aforementioned aspects, in several parts of the world the blockchain is being used to tighter the integration of ICT into power systems. Some new business models are being drafted and some market designs are changing more than others as presented in the end of Chapter 2. In Brazil, it still not known about similar approach, although little discussions have already been addressed<sup>1</sup>. This condition is one of the reasons for choosing this topic to be studied.

Moreover, the Brazilian DG legislation looks to be ahead of its time and linked to the blockchain purpose. Similarly to how Bitcoin has been allowing profitable trade between small service providers and customers, reducing the infrastructure cost to do so, the Brazilian shareable generation model (ANEEL, 2016a) is welcoming for such TE system. In this way, a blockchain application can ensure the active participation of customers on the power grid for a reliable buy/sell/donate transaction between members of a community, condominium, neighbourhood, municipality, or even different countries (COELHO et al., 2016a).

## The work purpose

Following this reasoning, this dissertation seeks to contribute with a quasi-proof of concept (POC) about the integration of the blockchain technology with the micro/minigrid power systems. It will be demonstrated how to support the power management inside Brazilian shared generation groups with a simple blockchain application algorithm (*smart contract*), that should be future expanded into a full distributed application (DApp).

However, it is not the task of this work to examine physical system operation and planning of both DGs and power grids, either introduce a market design to handle

---

<sup>1</sup> A fast search for the term “blockchain” at the Thesis and Dissertation Catalog of the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES) repository (available at: catalogodeteses.capes.gov.br/catalogo-teses) shows up the diverse of fields that have been considering this technology but so little for the energy sector in the country.

with supply-demand coordination. But this study has two main outcomes laying out the integration of blockchain with power systems: (i) overcomes the scalability issue; and (ii) supports a business model in existing market design.

The methodology comprises the understanding of the pillars of the TE paradigm and the comprehension of blockchain technology. Therefore, the study of the Brazilian legislative framework where micro/mini-grids lay down has delineated the boundaries of the application. Additionally, an overview of the blockchain structure has led the study through a detailed evaluation amongst the diverse platforms available for the development of the DApps. The investigations are not exhaustive but have enough bibliographic research to sustain the proposed distributed application and to conduct the quasi-POC regarding other use cases under development around the world. The methodology also follows a software engineering's standard to document the processes.

Finally, after the contextualization about the reason and goal of the current work, the following three chapters go deep into the above discussion. Chapter 1 seeks to present the current micro/mini-grid Brazilian scenario with a systematic view of its operations, and opportunities towards a TE context. Appendix A gives a little broader discussion about the Brazilian commercialization of electric power in order to place the DG approach in the whole sector chain. Chapter 2 introduces concepts related to the emerging blockchain technology. Principles about the types of available blockchains are discussed too, and some examples of applications on the power sector chain are also shown. Chapter 3 focuses on the proposal itself, with an analysis of the project requirements and the features of the chosen technology, the *Neo Blockchain*. Appendix B, Appendix C, and Appendix D complement the smart contract developed, which is replicated at full in Appendix E. They also support the results presented in the end of the concerning chapter. Section 3.7 concludes the essay reviewing the findings and limitations of the outcomes, giving suggestions for future improvements, and presenting the publications made so far.

## 1 THE BRAZILIAN DISTRIBUTED GENERATION

This chapter presents the Brazilian micro/mini-grid market design and aims to depict its categories in order to identify where the proposed distributed application (DApp) should be used for. Initially, the Distributed Generation (DG) is exposed in contrast to the country power generation size. Lately, the target micro/mini-grid category is slightly detached to give a good understanding about its operation. Important to highlight that throughout this chapter, all the terms will refer only to the Brazilian context.

As previously indicated, the DG has been considered an alternative power source to traditional big power plants. The latter primarily encompasses hydroelectric and thermoelectric power sources to sustain the country power needs. As a matter of curiosity, the National Power Grid System (SIN) has about 162 GW of power capacity (ANEEL, 2017a). The former mainly comprises photovoltaic solar panels to feed local power demands. However, it must be connected to the local power distribution grid (ANEEL, 2016a), and its power capacity installations define the boundary between microgrid, up to 75 kW, and mini-grid, up to 5 MW.

The SIN represents the national energy policy to guarantee affordable electricity for the whole country, while the DG is still a benefit for some consumers, even though the price of Distributed Energy Resources (DER) has been decreasing on the last 10 years (HAHN; MIGHELÃO, 2017). Although they target different interests, both applies for the mid/low-voltage consumers under the power distribution grid, classified as captive consumers<sup>2</sup>. This kind of consumers is sheltered by the Government with the lowest possible electricity tariff, a value determined through a methodology of price diversification based on forecasting different energy scenarios in the country. However, those with the DG have an additional option of energy price defined by their own investment in power generation.

Therefore, the captive consumers that generate energy have now another role in the distribution grid. They are known as *prosumers* (*producer + consumer*) and are grouped under a specific regulation of the electricity sector. Overall, there are four categories of prosumers that can be divided into two main groups based on their generation management type. One that is for self-consumption, and another for shareable consumption but always taking into consideration that the generation and consumption units must be inside the same distribution utility coverage area (ANEEL, 2016a).

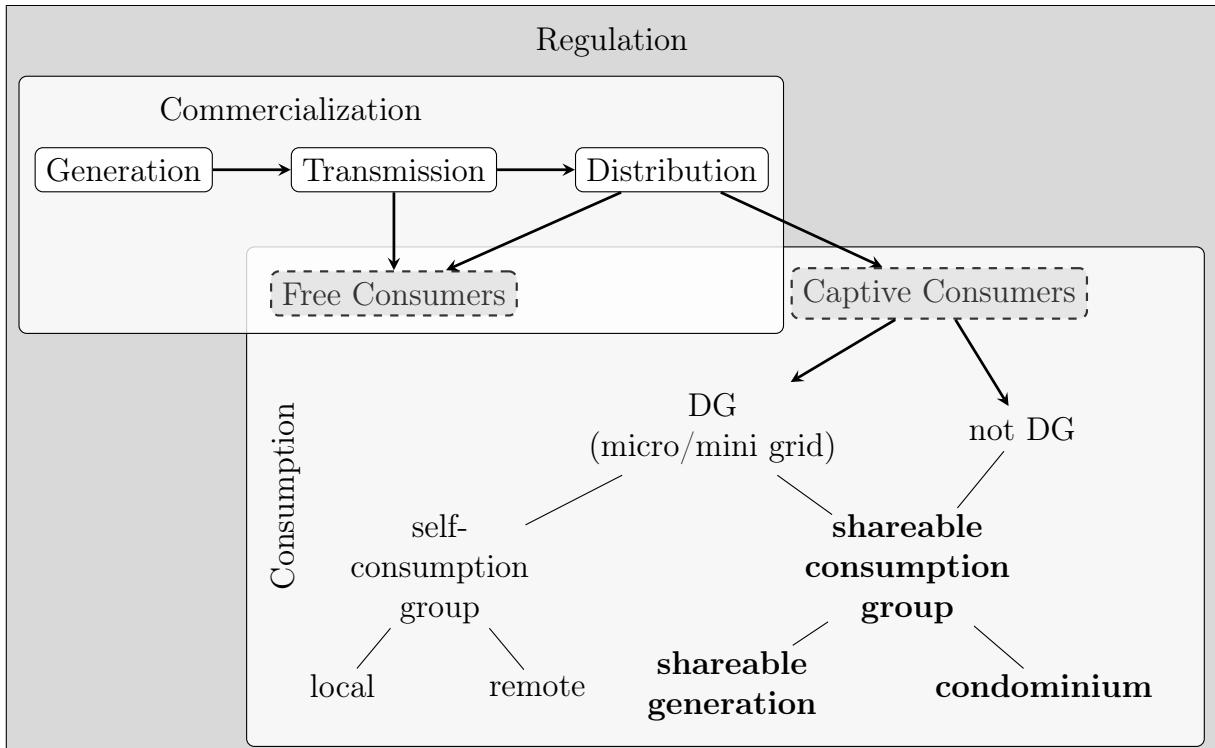
So, the self-consumption group is for individual use. The DG may be either on the same local of power consumption or in another place (namely *remote self-consumption*).

---

<sup>2</sup> For a better comprehension about the captive and free consumers consider Appendix A.

On the other hand, the shareable consumption group refers to sharing generation between customers, either they are from a condominium (known as *enterprises of multiple consumer units*) or from a consortium/cooperative (known as *shareable generation*) (ANEEL, 2016a; ANEEL, 2016b). Figure 1 summarizes this classification and contextualizes the DG into the power energy sector. Note that the unique place where different prosumers and consumers can be under the same regulatory framework is at the shareable consumption group specification.

Figure 1 - Classification of DG under the Brazilian legislation, and its positioning in the power sector chain.



Source: The author, 2019.

Even with different categories, all of them can “store” any energy surplus on the distribution grid for latter consumption with no additional cost<sup>3</sup>, instead to storage this energy on private batteries (ANEEL, 2016a). Although the delivery of the surplus electricity over those two groups has different forms, it can be modeled under a common principle, where the total energy ( $E_T$ ) is divided by the total number of participants. Equation 1 catches this with the number of participants ( $n$ ) ranging from 1, as for the self-consumption group, to multiple members ( $m$ ), as for the shareable consumption group.

$$E_T = \sum_{n=1}^m E_n \quad [\text{Wh}] \quad (1)$$

<sup>3</sup> Until the expected approval of changes in the *Resolução Normativa nº 482/2012* in the year of 2019.

For the self-consumption group, there is nothing special because the distribution of the surplus power follows the interest of only one customer. Even for the remote self-consumption case, the split of energy between multiple power units is up to one decision maker. However, specific rules guide how it will happen for the shareable consumption group accordingly with the members own agreement.

Usually, the cost and benefit of an investment in DG will be proportionally split among the participants. For instance, the condominium case has multiple consumer units and people to be taken into account. Thus, each member's contribution over the whole investment will represent her/his right over the energy surplus, i.e., an investment of 10% guarantees 10% of the electricity generated in a given period. This model represents the investment shares as percentage values, called as well by quotas.

So, Equation 3 complements the Equation 1 arranging the dual relationship of cost-benefit for a particular customer. Consequently, the electricity obtained by a member ( $E_m$ ) is proportional to the total power generation ( $E_T$ ) by a factor ( $q_m$ ) of her/his contribution ( $C_m$ ) to the DG investment ( $I$ ).

$$q_m = \frac{C_m}{I} \quad (2)$$

$$E_m = q_m \cdot E_T \quad [\text{Wh}] \quad (3)$$

The surplus energy discussed above is a measure known as Electricity Compensation System (ANEEL, 2016b). It is also referred as “energy credit” and can only be used for personal purposes, i.e., the surplus energy cannot be transferred for anyone else, even if part of a shareable consumption group. But a member's quota can be changed to reflect on lesser or greater credit on upcoming tariff-time periods. The way to do so is where the present work lands on.

Moreover, any modification on the shares of the surplus power generated must be known by the distribution utility in order to keep record of the energy credit that will be distributed. But beyond to what is exposed by Equation 1, the utility must get notes about member's quotas as well (Equation 2), because each member has a personal consumer profile, and consequently, a different energy credit balance.

A useful example of how this dynamic happens on each group is presented at Table 1. Considering only a tariff-time of two consequent months with few variable-changing conditions, the current example aims to encourage the thinking of the countless scenarios that may arise when the number of members scale up.

Table 1a is a typical situation for a person that has two homes but only one with DG. The energy share of each unit is proportional by a factor (quota) to the sum of the energy surplus. On the *month 1*, both paid a lower amount of energy but had no credit. However, on the *month 2* the quota has changed to satisfy only the *U2*. This unit has

gained credit because its consumption has also diminished but  $U1$  had to pay the full bill.

Table 1b is a non-realist size of a shareable consumption group but big enough to state some of its power management challenges. Similarly to the previous example, the generation is counted as a unit and later distributed proportionally by each member's quota, even though  $M3$  having its own power source.  $M2$  represents only a power site and has no consumption.  $M1$  benefits from the group generation by a factor of 40% but can negotiate it for better conditions since  $M3$  tendency to small power consumption, and consequently make credits, that will not be consumed in the future.

As was pointed out some paragraphs above, at Table 1a the decision to change the quotas of each consumer unit is up to one and only one person. She/He assumes all the risk for a bad decision. But Table 1b differs because of the number of members that it can integrate. Even for a small group with people known to each other, to keep confidence on the management of personal interest is dubious, and certainly, the growth of members will impact on the management scalability of the DG. Therefore, blockchain applications can bring transparency and security to these operations.

Table 1 - Example of the DG management for a given time-frame for both group classifications.

	Surplus [kWh]	Quota [%]	Share [kWh]	Consum [kWh]	Payable [kWh]	Credit [kWh]
<b>month 1</b>						
U1	0	25	12.5	80	67.5	0
U2	50	75	37.5	80	42.5	0
<b>month 2</b>						
U1	0	0	0	80	80	0
U2	50	100	50	40	0	10

(a) Self-consumption group.

	Surplus [kWh]	Quota [%]	Share [kWh]	Consum [kWh]	Payable [kWh]	Credit [kWh]
<b>month 1</b>						
M1	0	40	52	80	28	0
M2	100	0	0	0	0	0
M3	30	60	78	80	2	0
<b>month 2</b>						
M1	0	40	52	80	28	0
M2	100	0	0	0	0	0
M3	30	60	78	50	0	28

(b) Shareable consumption group.

Source: The author, 2019.

## 2 GENERAL CONCEPTS ABOUT BLOCKCHAIN APPLICATIONS

The blockchain technology has emerged in the context of Bitcoin crypto-currency as a proposal to solve the double-spending problem presented in a transaction of digital money, i.e., the capability to exchange the same amount more than once (NAKAMOTO, 2009). For instance, it is like if someone uses the “same” cash in different places because the accuracy of fake money copies don’t call attention until its go back to central bank, where identifiers are checked.

Independently of the kind of money, the current veracity analysis for any transaction between stakeholders is made by a third-party trustful by them. Its a common approach which uses an impartial stakeholder capable to certify each transaction, and to store the history of exchanges in a database, also called ledgers. This method of data management increases transaction costs, that limits minimum practical values and small deals, however, it guarantees the action of reversal transactions when needed (NAKAMOTO, 2009).

Thus, the Bitcoin solution was a protocol that uses a P2P network architecture with a controlled redundancy of transactions registrations in a write-only format verified by each user in its network in order to allow “transactions that are computationally impractical to reverse [that] would protect sellers from fraud, and routine escrow mechanisms [...] to protect buyers” (NAKAMOTO, 2009, p.1).

But the blockchain technology goes beyond money applications, it can be used to track any asset transaction, and it represents a new form of information management that can potentially impact the whole society. Since the trustful intermediary is indeed the blockchain network, where users agree upon how it operates to reach a consensus about the validity of a transaction (SWAN, 2015), several blockchain technologies have arisen to fit all kinds of business (KANDASWAMY, 2016).

However, due to its initial stage, there are some mismatched terminologies. For instance, the word *blockchain* has been using to the data structure (XU et al., 2017) reproduced by and stored at nodes, i.e., the ledger registrations. And the cryptographically decentralized network connecting each node, represented by a P2P communication overlay network with a consensus method is referred to *blockchain technology* (XU et al., 2016), a type of *Distributed Ledger Technology (DLT)* (PERLMAN, 2017; UK Government Chief Scientific Adviser, 2016). Therefore, some standards are under development to cope with it and to define the better use and compatibility between companies and services, such as the technical committee *ISO/TC 307 - Blockchain and distributed ledger technologies*<sup>4</sup>,

---

<sup>4</sup> More information in the webpage: [www.iso.org/committee/6266604.html](http://www.iso.org/committee/6266604.html).

the ITU-T Focus Group on *Application of Distributed Ledger Technology (FG DLT)*<sup>5</sup>, and the IEEE Standards Association (IEEE-SA) through the *IEEE Blockchain Initiative (BCI)*<sup>6</sup>. The latter already has active standards projects on TE systems<sup>7</sup>, and for the energy sector<sup>8</sup>.

Briefly, two analogies portray well what the blockchain technology is for the business application point of view. One can be seen as a big digital ledger shared by all those who participate in the system, in which transactions are irreversibly recorded. It is the chronological record of all transactions compiled and validated that occurred in the network. It is unique and shared by a specific system (ENDEAVOR, 2015). And the other can be said that the blockchain is like a reef of coral in which only the last millimeters represent active biomass, the rest is only a dead image of the past and accessed only on rare occasions to check historical data (MERZ, 2016).

As noted in the Introduction, the blockchain technology components are not new if analyzed separately. Distributed networks, and public and private keys cryptography have been part of our daily life for years, however the novelty is in how to generate and communicate consensus on a common redundant database updated through a decentralized network (ALEIXO, 2016). This behaviour and the main blockchain terms are described in Section 2.1. Thereafter, Section 2.2 exposes a brief evolution of the blockchain technology. Following, some relevant blockchains are presented in Section 2.3, as well as some of its astonishing DApps in the energy sector in Section 2.4.

## 2.1 Terminology and characteristics

The blockchain technology is better understood if its protocol is analyzed by two views. As a data structure, it is an ordered list of identifiable blocks that points to a previous block in the chain (XU et al., 2016). However, a block has only hash values that represent several “real data files” stored on somewhere else. Until creating the hash that identifies the block, some hash functions are individually performed on each file, so they can be grouped under a fixed-length value. It also makes the protocol tampering-proof (WOOD, 2016).

---

<sup>5</sup> More information in the webpage: [www.itu.int/en/ITU-T/focusgroups/dlt/Pages/default.aspx](http://www.itu.int/en/ITU-T/focusgroups/dlt/Pages/default.aspx).

<sup>6</sup> More information in the webpage: [blockchain.ieee.org/standards](http://blockchain.ieee.org/standards).

<sup>7</sup> More information about the P825 – Guide for Interoperability of Transactive Energy Systems with Electric Power Infrastructure (Building the Enabling Network for Distributed Energy Resources) in the webpage: [standards.ieee.org/project/825.html](http://standards.ieee.org/project/825.html).

<sup>8</sup> More information about the P2418.5 – Standard for Blockchain in Energy in the webpage: [standards.ieee.org/project/2418\\_5.html](http://standards.ieee.org/project/2418_5.html).

As a P2P network, it inherits some architecture functionalities, such as (i) the peers can leave and join the network anytime (NAKAMOTO, 2009; PACITTI; AKBARINIA; EL-DICK, 2012), (ii) the presence of some “special” peers with distinguish characteristic (eg. processing power) in order to “perform complex functions such as indexing, query processing, access control, and meta-data management” (PACITTI; AKBARINIA; EL-DICK, 2012, p.3), (iii) each peer is both a client and a server which are normally convenient for large-scale applications (PACITTI; AKBARINIA; EL-DICK, 2012), (iv) an overlay network designed to have scalability with low costs, without compromise peer autonomy (PACITTI; AKBARINIA; EL-DICK, 2012), and (v) the growth of the network resource and content availability is proportional to the number of peers, but the network keeps an invariable response time, and a high search throughput (PACITTI; AKBARINIA; EL-DICK, 2012).

However, they keep apart on the manner the network is used by its applications. While traditional P2P networks share computer resources (content, memory, storage, processing, bandwidth, and so on) to achieve scalability for content distribution (PACITTI; AKBARINIA; EL-DICK, 2012), the blockchains achieve the same requirement replicating the data structure across the peers to guarantee interoperability. Although good for data retrieving and reliability, it may represent a limit on the file data size and on the data structure.

In addition, the participants of this network can be split into two node types: (i) one that just uses the network as a service to communicate directly with peers, to contribute to a transaction verification and to store information, and (ii) another (the “special” peer) that works to keep the network organized, appending new blocks of information on the ledger. The latter is responsible to integrate the consensus method about the new information to be spread and updated across the network.

What concerns for data protection, every node has two *cryptographic keys* to keep its connections reliable. While the public key is for information traceability, the private key is used to certify the data ownership in a transaction.

The *transaction* is indeed any message about some arbitrary operation in the network. For instance, it can be an exchange between peers, an entity registration, or yet the creation of crypto-currencies. It is supported by the so-called *smart contracts*, that just minimize the need for a trustful third-party to solve the same common problem (SWAN, 2015). Its arbitrates an agreement between peers, being basically a business logic running on the blockchain (HYPERLEDGER, 2018a).

However, besides businesses logics written by ordinary programmers, each blockchain technology has its particularity to handle with all sort of contracts. This establishes two different types of smart contracts. One of them acts on the network nodes validators. They are pre-build smart contracts with essentials features installed to deal with all the network functionalities. The other is the on-chain smart contract, which follows business

purposes and are deployed as a transaction. When successfully appended on the ledger, their codes are part of the network and available to be called by subsequent transactions (HYPERLEDGER, 2018a).

The relationship between these two smart-contracts defines the blockchain system behaviour and how an on-chain smart contract are processed in front of possible errors and validations. Briefly, an on-chain smart contract can be divided into three parts: (i) the inputs, that gather the contract identifier, the transaction request, any dependencies that may exist, and the current state of the ledger; (ii) the interpreter, that has the information about the ledger current state and the smart contract code itself; (iii) the outputs, that have the new transaction state (accepted/rejected) and any side effects, as a notification of something (HYPERLEDGER, 2018a).

Therefore, when a smart contract transaction is processed, the inputs are immediately checked by the interpreter in order to reject any invalid request. The outputs are generated accordingly with the verification, resulting in different values if the request was valid and accepted, or has raised any errors. From here, the upcoming steps are handled by the installed smart contracts, i.e., this depends on the particular behaviour of each blockchain technology. They are responsible to define what has to be done with the transaction based on the output. If the transaction is ready to be appended in the ledger, the consensus service is called, otherwise the error service is called (HYPERLEDGER, 2018a).

Independently of the error type, the crucial feature to be considered is how the blockchain deal with the new state produced by a smart contract transaction. Because its immutability feature may establish a restriction at execution time that cannot be directly overcome like a conventional application does (PERLMAN, 2017).

Furthermore, as a code infrastructure governing the exchanges, the smart contracts can also be fully automated to be triggered when specific future event happens in a certain time frame (SWAN, 2015; XU et al., 2017), taking advantage of lower costs for contracting, enforcement, and compliance (PERLMAN, 2017). However, the trigger feature is not so simple to implement due to the monitoring condition of the blockchain status may rely on off-chain mechanisms, and because of the personalized structure of each smart contract.

To complement a transaction process, nodes must reach *consensus* about the veracity of the information to be appended in the network. An automated distributed mechanism is used to regulate (i) the criteria new items should meet to be added, (ii) how the incentive scheme works for the (“special”) peers responsible for that, and (iii) how the possible conflicts are solved (XU et al., 2017). Thus, consensus must satisfy the properties of: (i) *safety* because it has to guarantee that each node has the same output for the same sequence of inputs, i.e., keep the system behaviour equal for every node when a change may occur in a given node; and (ii) *liveness* since each non-faulty node must ultimately receive every submitted transaction in the absence of communication troubles

(HYPERLEDGER, 2018b).

This is an odd characteristic of blockchain systems, which indeed is used for classifying a range of DLT's. Therefore, some kinds of algorithm that integrate the whole consensus mechanism are described below:

**Proof of Work (PoW)** is a random process to discover a transaction hash number (PWC, 2016; XU et al., 2016). It is a hard task to accomplish with, so nodes with high processing capacity has advantages front of others (XU et al., 2016). Additionally, some conditions to satisfy the system latency and speed increase the task difficulty (PWC, 2016). It is not for nothing that those “special” nodes are called “miners” (PERLMAN, 2017). Right after the hash value has been found, the process is validated throughout every node. When the majority of nodes has done it, the transaction is confirmed to the counterparties, because most of its outputs have already been appended in the ledger (PWC, 2016). PoW is very costly to produce but easy to be verified, and the transaction processing rate is limited by the network consensus rules (XU et al., 2016).

**Proof of Stake (PoS)** simplifies the PoW method by distributing the verification process between peers proportionally to their shares of the network (PWC, 2016). So, if a peer has 10% sharing of the total blockchain assets, it will have to deal with 10% of the mining process. This approach reduces the complexity, the energy use and the operating costs of an entire transaction step (PERLMAN, 2017). Due to its openness to being part of the block generation network relies on the share of financing, the crypto-currency process uses to be referred to as “minted” (PERLMAN, 2017). In addition, at PoS blockchains, any malicious attack would require a large amount of currency to work out, which is very expensive (XU et al., 2016).

**Proof of Elapsed Time (PoET)** is another alternative to PoW developed by Intel that is a hybrid of a random lottery and first-come-first-serve basis algorithm (LINUX FOUNDATION, 2017). Basically, the validator peers receive a random wait time to execute a given request. The one with the shortest wait time wins the dispute to create the next block on the chain (LINUX FOUNDATION, 2017; INTEL, 2018). The PoET uses new secure CPU instructions, as a Trusted Execution Environment (TEE)<sup>9</sup>, to ensure the safety and randomness of the peer election process as an alternative to the costly investment of power and specialized hardware (INTEL, 2018).

---

<sup>9</sup> It is an isolated and secure CPU area with confidentiality and integrity performance. A general information about it is available at: [en.wikipedia.org/wiki/Trusted\\_execution\\_environment](https://en.wikipedia.org/wiki/Trusted_execution_environment).

**Practical Byzantine Fault Tolerance (PBFT)** is one of the algorithms used to solve the Byzantine Generals Problem<sup>10</sup>, i.e., to keep a distributed networking reliable and functional in the presence of failures of any nature (hardware or software). Although the Byzantine Fault Tolerance mechanism is a universal solution for system communication interferences, and all the aforementioned consensus methods are considered solutions to this problem, the following protocols are optimizations of the PBFT only (NEO, 2018; LINUX FOUNDATION, 2017). However, either the **Simplified Byzantine Fault Tolerance (SBFT)** (LINUX FOUNDATION, 2017) and the **Delegated Byzantine Fault Tolerance (DBFT)** (NEO, 2018) have a single validator, known and trusted by all peers in the network, responsible to append new blocks in the ledger. And the consensus is a result of the interaction of some other nodes ratifying the truthfulness of the transactions. In the end, the process must comply with the consensus of  $2f + 1$  nodes in a system with a total of  $3f + 1$  nodes, where  $f$  is the number of faults. Thus, both keep good performance front of the network scalability.

**Proof of Authority (PoA)** turns the nodes allowed to create new blocks and append them on the ledger identifiable, making their personal information available for cross-reference. Based on their responsibility in the network, they are called “authorities”, and there are no need of mining to reach consensus, because each authority is carefully chosen and rewarded to validate the transactions on behalf of the network interests (LINUX FOUNDATION, 2017). At the PoA the peer reputation is enough to keep the network trustworthy.

Summing up, the different ways to get consensus stand on different network resources and fault tolerance models, which can be through the use of lottery-based algorithms – PoET and PoW –, or through the use of voting-based methods – PoS, PoA and PBFT optimizations (HYPERLEDGER, 2018b). The former kind of algorithms has a good scalability factor (LINUX FOUNDATION, 2017), although PoW does not perform well for the speed and finality of the consensus, i.e., a transaction verification last long and a fork on the network may invalid a given transaction. Otherwise, the latter kind provides low-latency finality<sup>11</sup>(LINUX FOUNDATION, 2017) but not so good scalability performance, because the trade-off between these both specifications relies on the number

---

<sup>10</sup> The article *The Byzantine Generals Problem and Blockchain Consensus Model Proof of Work, A Deep Dive* presents a good discussion about it. It can be found and read at: [achainofblocks.com/2018/08/09/byzantine-generals-problem-proof-of-work](http://achainofblocks.com/2018/08/09/byzantine-generals-problem-proof-of-work).

<sup>11</sup> This means to append a new data on the ledger happens really fast. It is worth reading the article *Latency and finality in different crypto-currencies*, available at: [hackernoon.com/latency-and-finality-in-different-cryptocurrencies-a7182a06d07a](https://hackernoon.com/latency-and-finality-in-different-cryptocurrencies-a7182a06d07a).

of nodes responsible to append data in the network.

Finally, the Initial Coin Offering (ICO) is an alternative mechanism to distribute crypto-currencies (or tokens) for users of a given blockchain. Its usually represents a crowdfunding for a DApp development, either because of the way tokens are distributed by fundraising, or because of the predefined mechanism to generate the tokens (PWC, 2016), typically a different process than the one used to create native crypto-currency. Moreover, the distinctive token lets the ICO's representative weights the coins value from another crypto-currency, instead of relying on fiat-money.

### 2.1.1 Basic properties

In summary, the use of blockchain technologies is evaluated in contrast to current available solutions. In general, the design of any computer network must meet technological and social compliance. The former includes connectivity features, and flexibility to integrate with future changes. The latter refers to accessibility by humans with different levels of skill (PETERSON; DAVIE, 2011).

Therefore, both DLTs and traditional P2P systems have some advantages in relation to client-server structure, such as the properties of scalability (cost-effective with increase on use), autonomy and dynamic behaviour of peers, self-organization, decentralization, and fault-tolerance (reliability) (PACITTI; AKBARINIA; EL-DICK, 2012). But blockchains in particular add the properties of immutability (write-only), non-repudiation (irreversible), integrity (robustness), transparency, and equal rights (XU et al., 2017). Although they complement each other under a protocol specification, there is a fine line between what stands for blockchain applications and for network infrastructure.

In addition, some of those properties may be adjusted to give rise to different blockchains. For instance, the scalability factor and latency between submission and confirmation of a transaction are impacted by the consensus chosen (XU et al., 2017). Or even a different arrangement of data privacy access can allow new options to read the ledger but do not impact on the basic properties of the network.

The requirements across industries and business use cases that differently combine these specifications are numberless, which is a unique optimization opportunity for the technology (HYPERLEDGER, 2018b). Further analysis about it is presented at Section 2.3, that is also used to define the blockchain chosen on Chapter 3.

## 2.2 Timeline

The blockchain technology has been evolving since Bitcoin appearance, which has constituted the first generation of the DLT, remarkable by the decentralization of money and payments. Succeeding, the second generation stands for the decentralization of markets in general, because the transactions can be made with any kind of asset. At this time, a good understand of smart contract behaviour guides actions towards current models replacement, that supports the development of *DApps*. Last but not least, it is expected from the third generation a completely autonomy management system with smart contracts, a perspective capable to revolutionize the market due to its complexity (SWAN, 2015).

Nowadays, we are experiencing a variety of DApps ranging from online games<sup>12</sup> to popular initiative bills support<sup>13</sup>. Although it lacks formal definition, in the most cases, the DApps are open source platforms to work at, dependent of fee to be executed, and controversial about code adaptability for future adjusting. However, there are consent that a DApp is one or more smart contracts that runs in a decentralized network securely protected with a special feature of distributed storage functionality and data management (SWAN, 2015; PWC, 2016). In addition, the DApps ecosystem are vast with no clear target market defined by service providers, which results in different applications categories beyond the DLT classification development, i.e., a given blockchain framework used for breakthrough a food supply chain system can also fit a grocery store payment system demands (KANDASWAMY, 2016).

## 2.3 The blockchain's varieties

The projects with blockchain vary in its value proposition according to the market purpose, i.e., if it aims to manage operations between enterprises, or within a single company (KANDASWAMY, 2016). In addition, the degree of integration with a mechanism of payment processing should be considered as well. If the project is only intended for data communication or for synchronization of participants' applications, there are no special requirements. However, if the participants' interactions are used for accounting purposes,

---

<sup>12</sup> A good analysis of blockchain games is available in the article *Examples of Blockchain Games (and how they work)* at: [medium.com/crowdbotics/examples-of-blockchain-games-and-how-they-work-7fb0a1e76e2e](https://medium.com/crowdbotics/examples-of-blockchain-games-and-how-they-work-7fb0a1e76e2e).

<sup>13</sup> The application *Mudamos* strengthens the relationship between Brazilian voters and their representatives collecting their electronic signatures to enforce law changes. More information in Portuguese at: [www.mudamos.org](http://www.mudamos.org).

this can be done defining different levels of data access and share (MERZ, 2016). In other words, the type of blockchains can be categorized as the way its users agree on how to reach consensus (PERLMAN, 2017).

Therefore, the blockchain technology has been designating as *permissionless* and as *permissioned* (PERLMAN, 2017; PECK, 2017). Although both are “similarly designed for rapid detection of unauthorized changes to the data” (PERLMAN, 2017, p.8), the latter aims to overcome some challenges presented on the former (PECK, 2017). At permissionless blockchains, the consensus process and the access to read the ledger are open to everyone, i.e., the DLT has no owner. That’s why it is also referred to as public blockchains, keeping the attributes of decentralization, no censorship, no counterparty exposure, and an open, global membership (PERLMAN, 2017).

However, at permissioned blockchains, participants may be preselected, and the process of read and write the ledger may be restricted for different users (PERLMAN, 2017). It is usually split into “permissioned public” and “permissioned private”. The latter may have one or many owners, and the consensus method is limited by someone. The former configures as a trade-off in between the latter and the permissionless blockchains. Even though the latter is more restrictive, it allows a more efficient way to append data and faster verification processes, which means a direct correlation among the trust factor, the computational power required for consensus, and the speed of a transaction (PERLMAN, 2017).

Accordingly with Perlman (2017), there is a belief that this categorization may be more granular. For instance, a permissioned private blockchain, also referred to as Blockchain as a Service (BaaS), can be easily compared with a secure append-only database with similar “functionalities as a standard cloud-hosted application, with suitable access control and identity regime, that records specific actions of those involved in an appropriate ‘secure’ database” (SINGH; MICHELS, 2017, p.8). However, it lacks the sense of self-organized community in a trustfulness environment (PECK, 2017), even though being able to develop a new cloud computing marketplace based on the decentralized approach proposed by blockchain systems, such as the case of iExec<sup>14</sup>.

Based on the work from Xu et al. (2017), it is possible to understand how the elementary specifications of blockchains compose some of the systems available in the market. The publishings from Peck (2017, p.35) and UK Government Chief Scientific Adviser (2016, p.19) complement the aforementioned view with decision flowcharts to determine if a given application should be based or not on a blockchain. In summary, both agree that the proposed solution requirements do go beyond what a traditional database technology current offers, then defining what path for a kind of blockchain to

---

<sup>14</sup> More information in the webpage: [iex.ec](http://iex.ec).

follow based on ones ability to update the data, and trusty factors between counter-parties about immutability, reliability, data access and consensus compliance. Xu et al. (2017, table II) also presents a trade-off analysis about what data and computation should be placed on- and off-chain considering the cost efficiency, the performance, and the number of failure points that may impact on a design decision.

Table 2 - Systems design decisions regarding (de)centralisation.

Decision Design	Option	Impact			
		Fundamental properties	Cost efficiency	Performance	# Failure points
Fully Centralised	Services with a single provider (e.g. governments, courts).	⊕	⊕⊕⊕	⊕⊕⊕	1
	Services with alternative providers (e.g. banking, online payments, cloud services).				
Partially Centralised/ Partially Decentralised	Permissioned private blockchain with permissions for fine-grained operations on the transaction level (e.g. permission to create assets). It resembles to BaaS.	⊕⊕	⊕⊕	⊕⊕	*
	Permissioned public blockchain with permissioned peers (write), but permissionless normal nodes (read).				
Fully Decentralised	Permissionless blockchain, i.e., public blockchain.	⊕⊕⊕	⊕	⊕	Majority (nodes, power, stake)
Verifier (consensus method)	Single verifier trusted by the network: external verifier signs valid transactions; internal verifier uses previously-injected external state (PBFT optimizations).	⊕⊕	⊕⊕	⊕⊕	1
	M-of-N verifier trusted by the network (PoW).	⊕⊕⊕	⊕	⊕	M
	Ad hoc verifier trusted by the participants involved (PoS, PoA).	⊕	⊕⊕⊕	⊕⊕	1 (per ad hoc choice)

Legend: The design impact on essential properties ranges from less favourable (⊕), through neutral (⊕⊕), to more favourable (⊕⊕⊕) options. Also, the mark \* indicates that the number of failure points depends upon the design chosen, ranging from the ends of 1 to majority points.

Source: Adapted from Xu et al. (2017).

Table 2 summarizes this discussion about different informational system designs comparing their weaknesses and strengths. Besides the two big features of the resources access and the protocol (consensus method) of the network, which are organized in the rows and separated by the gray space, there are the key characteristics that impact the system, which are organized in the columns. The fundamental properties are the particular blockchain features previously cited at Section 2.1.1 – immutability, non-repudiation, integrity, transparency, and equal rights. These properties increase the impact on the decentralised design because they are the key variables of this system, and the opposite

is also true. Similarly happens with cost efficiency and performance, which are a direct consequence of the number of failure points available to keep the system reliability, i.e., in a system with alternative service providers, the cost is distributed and the performance enhanced. On the other hand, the row of the verifier statement represents the possibilities available to reach consensus about a certain transaction, i.e., if the consensus protocol is a lottery-based or voting-based algorithm-kind. Now, the fundamental properties correspond to the network infrastructure – scalability, low-latency and finality – and is directly related with the consensus protocol used to govern the network, which also reflects in the number of failure points. The cost efficiency and the performance follows the economies of scale methodology of each consensus behaviour.

Note that the separation between the network classification to the consensus types evidences the potential range of blockchain configurations. Moreover, permissionless and permissioned blockchains can be identified too, and they are not related to a specific consensus method. Indeed, a system may have some components or functions decentralised while others are centralised (XU et al., 2017). For the sake of simplicity, some advocates to split those definitions into accessibility and visibility levels. The former is related to permissioned-permissionless conditions and expresses who can write to a blockchain. The latter refers to public-private duality and stands for who can read from a blockchain (HUSEBY; PIEKARSKA, 2018; BEHLENDORF, 2019).

### 2.3.1 A framework of reference

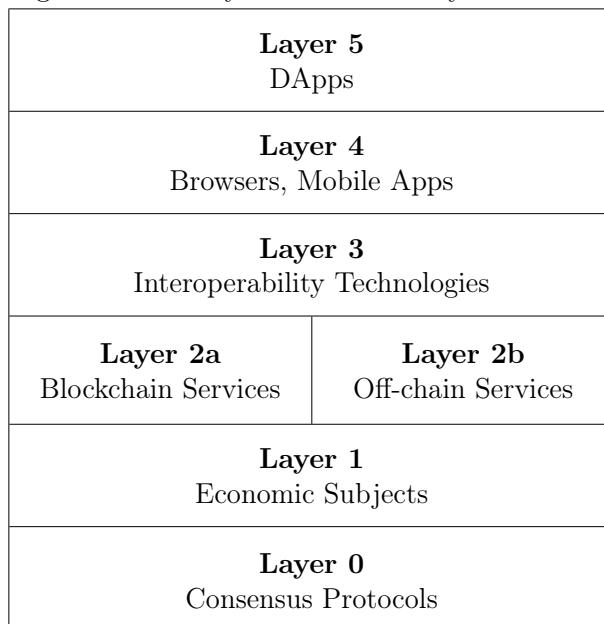
As beforehand stated, blockchains have been implemented to accomplish with different business purposes, which makes hard to distinguish the features that stands for the application and for the distributed system itself. However, a modular design pattern common for all of them is useful to identify and compare their functionalities. This approach seeks to allow an intercommunication of the diversified DLTs towards the third generation of blockchains (BUTERIN, 2014). Figure 2 shows a suggestion from Buterin (2014) of the aforementioned layering design. Although some stages may be directly linked at a given blockchain designation, the figure captures another point of view of distributed system properties.

In the first instance, the layer 5 and the layer 0 used to be the ones to choose the right DLT platform, mainly influenced by the concepts presented on the Table 2. However, the former guides the business attributions needed to create a valuable DApp, which makes any analysis a bit more specific to the features available by the latter in order to fulfill the requirements arranged through other layers. For instance, the considerations for the development of smart contracts and identity services, which can be provided either on-chain (layer 2a) or off-chain (layer 2b), are both very close to the consensus layer decision

(layer 0) (HYPERLEDGER, 2018a).

Thereby, the consensus protocols layer (layer 0) must allow services to interface with the application layer (layer 5) over (i) economic subjects (layer 1) as game theory to thrive at the use of crypto-currencies; (ii) data provision (layer 2) as part of smart-contracts and the blockchain indeed (layer 2a) or as services to search, query and trigger an action (layer 2b); (iii) interoperable services (layer 3) between platforms by means of best practices and standards for the development of “networking layers, cryptographic algorithms and other low-level components”; and (iv) usability (layer 4) with current and future user interface platforms (BUTERIN, 2014).

Figure 2 - The layers of an ordinary blockchain.



Source: Adapted from Buterin (2014).

## 2.4 The distributed applications in the electricity sector

The unique characteristic of the blockchain technology architecture by privacy levels has been allowing the development of several new applications accordingly with specific needs. For some purposes, its use can be interpreted merely as a cloud computing service improvement, and the DApps have the capability to provide a better use – and a rapid experimentation – of the system with full abstraction of its behavioural integration on the blockchain network (SINGH; MICHELS, 2017).

As was mentioned in the previous chapter, the electricity network used to have a top-down business model with utilities and big power generators sending electricity to customers, however, the market relationship varies by where it is located. In one hand, directly trade of electricity between generators and residential consumers is already a

reality in the European Union (EU) (MERZ, 2016; MEDEIROS, 2016). On the other hand, in Brazil its mechanism is limited by the level of power consumption, and not by consumer class. Therefore, the technology configuration range capability allows the blockchain to be used differently throughout the whole energy sector chain, as synthesizes the Table 3.

In this context, new technologies to provide the exchange of information and to allow interactions between different autonomous agents, embedded with artificial intelligence tools and mechanisms such as blockchain, should be a feasible solution and a trend for the next few years (COELHO et al., 2017). Moreover, the innovative approach of DLTs enable diversified experimentation models of applications, such as the case with DG, mainly with decentralized solar energy networks (RADU; ROXANA; ALEIXO, 2015).

As a result, the blockchain projects can overcome the challenges with decentralized energy systems optimization, and even prepare customers for new approaches on the electricity sector that may arise, such as real-time pricing. For instance, they can enhance the consumers' consciousness about their power consumption, the prosumers knowledge about their generation request, and even social awareness about their impact on the power

Table 3 - Blockchain applications in the electricity sector chain.

	Generation	Transmission	Distribution	Trading	Sales	Metering	Other areas
<b>B2C energy trading / peer-to-peer systems</b>							
Microgrids (peer-to-peer)	•	•	* (5)		•	•	
Grid management systems	•	•	•			•	(7)
Trading based on blockchain/smart contracts			(6)	(4)(9.2)			(7)
<b>Mobility</b>							
Charging process management/payment							(8)
Charging station handling							(8)
Ride sharing							•
<b>Asset management</b>							
Data collection/integration for individual assets (“provenance”)	(1)						
<b>Other energy use cases</b>							
Certificate handling (e.g. for renewable energy usage)				•	•	(9.1)	(3)
Payment of invoices with crypto-currencies			(2)	•	•		(3)
Blockchain-based supplier switching management			•		•		(9.3)

Legend: The notation • refers to the areas where the blockchain applications are being developed, and the current study focus is indicated by the \*. The notation (n) refers to the projects identified by the current study, where n represents the respective project described in Section 2.4.1.

Source: Adapted from WEC (2017, fig. 3).

grid (PECK, 2017; WEC, 2017). Therefore, for the Brazilian case, prosumers could leave a position of just accumulate energy credits to an active position, in which the excess power generated could be converted into profit through the concept of TE demonstrated before in the example at Chapter 1.

From this premise, worldwide solutions are being developed to enable the active participation of consumers in the electricity market, as shown on Table 3, and described in the following text segment.

#### 2.4.1 New electricity market examples

- (1) In a more simple and international way, the *SolarCoin* is a digital asset that rewards owners of solar power generation, being basically a technology to encourage decentralized, clean, and renewable energy generation, which aims to reduce the payback time for the solar installations (DCI, 201-). Its crypto-currency *Solarcoin (SLR)* works similarly to credit cards miles program, 1 MWh is equivalent to 1 SLR. It is freely issued into the network “in exchange for proof of solar energy production”, and it is also “a reward currency protocol based on a low-carbon blockchain” indeed (GOGERTY; JOHNSON, 2018, p.12).
- (2) On the other hand, the Marubeni company is allowing Bitcoin payments options for electricity consumption in some regions of Japan (GROARKE, 2016). While a real case application in South Africa developed by Grid Singularity uses Bitcoin transfer for accounting clearing of electricity pre-paid system (HESSE, 2016).
- (3) In Dhaka, Bangladesh, the *ME SOLshare Ltd* is a social enterprise that offers P2P trade system for solar power generation, and finance photovoltaic installations for low-income families. Founded in 2014, the SOLshare operates off-grid electricity in rural areas and allows its users to have an income directly from the energy of the Sun. Its goal is to empower people to become entrepreneurs, enabling the creation of a bottom up smart grid, and being ready for the future integration to the power network of the country<sup>15</sup>.
- (4) In a similar approach, the Alliander utility aims for “real-time” energy trade on the Island of Texel (Holland) with smart meters linked to blockchain technology. There, a new business arises towards the wholesale market (GROARKE, 2016).

---

<sup>15</sup> More information in the webpage: <https://www.me-solshare.com>.

- (5) In the case of the *Brooklyn Microgrid*, it is still limited to some consumers in Brooklyn, New York. The owner of a PV power can sell its generation to a neighbor using a smart contract available by the startup LO3 (MERZ, 2016). But this model still faces regulatory issues of energy transaction, since it is under the concession area of different distribution utilities (MENGELKAMP et al., 2018a), which can not occur either in Brazil as stated by national DG legislation. However, this POC has brought the discussion of the feasibility of local energy market place, in which market design should be considered, as well the economic evaluation of the market mechanism for the trading between consumers and prosumers. Nonetheless, some questions still require deeper investigation to consider blockchain as the main ICT for the system under consideration (MENGELKAMP et al., 2018b).
- (6) In Australia, *Power Ledger* goes a little further into the P2P interaction, allowing the transaction between the units of a building with other consumers of the distribution grid, thanks to its tight relationship with the local utility. Owners of a DG can decide for who they want to sell their surplus energy and at what price. In the platform provided there is a mechanism of negotiation and clearing that is transparent, auditable and automated in the benefit of prosumers and consumers (POWER LEDGER, 2017). They have already expanded to Auckland area, New Zealand, with expectations that schools, community groups and residential houses participate actively in the initiative (GROARKE, 2016).
- (7) Other actions are taking place on different uses of electricity as well. For instance, the *Wien Energie* utility uses the distributed technology to optimize and to save costs of gas trading for power generation (GROARKE, 2016). Established in Austria, the system is used to guarantee the stability of the grid in case the Sun or the wind do not attend necessary demand (WIEN ENERGIE, 2016).
- (8) And in Germany, the *RWE* together with the company *Slock.it* uses the blockchain to manage electric vehicle recharge in public charging stations. They use an accounting unit supported by different energy suppliers to provide vehicle drivers with a standard method of payment. The RWE system is based on the product *BigchainDB* by Ascribe from Berlin, but it is not yet known how smart contracts are used for unlocking charging stations (MERZ, 2016). A similar solution is presented by BlockCharge, where the idea about how it works is explained in the following video ([youtu.be/0A0LqJ9oYNg](https://youtu.be/0A0LqJ9oYNg)).
- (9) Moreover, real market applications have been tested up by the CoLab, an IDEO's hub for collaborative innovation, that designs human-centered projects. They have built three prototypes to understand blockchain potential with electricity applications. The Smart Solar (9.1) directly connects a solar panel to the blockchain

network in order to tracks its generation, and automatically issue a personal digital Renewable Energy Certificate (REC). The Shift (9.2) is a marketplace to trade energy and a self-management device to operate under power rates flexibility. And the Plug 'n' Paid (9.3) is the power device manager for homes, which uses Artificial Intelligence (AI) to adjust preferences and behaviours of homeowners looking for power efficiency. It also manages consumption, buys power in real time, and trades power with neighboring homes accordingly with pre-charge payment.

### 3 THE PROPOSED DISTRIBUTED APPLICATION

As was pointed out in previous chapters, the advent of distributed applications (DApps) in the whole electricity sector chain has been strengthening the integration of ICTs into the power network. Despite the technology used so far has been enough to keep the infrastructure working through third-party services, the blockchain enables a window of wisdom at the population level, where there is a big amount of low power customers mainly fed by one source, the distribution utility.

Moreover, aware that the Brazilian Distributed Generation (DG) legislation is inclined to the blockchain business model, and the expected expansion of Distributed Energy Resources (DER) as a consequence of consumers looking for lower tariffs and renewable power energy, the application purpose is to allow the trade of electricity within a community that shares the responsibility of managing their own power generation to supply their own power demands without relying on a third-party to do so.

The community scope is the group of shareable consumption, i.e., enterprises of multiple consumer units (condominiums) or shareable generation (consortiums or cooperatives). The members of the community range from power consumers to prosumers. Although they can't sell back their surplus generation to the power utility, they can determine the rules to transfer electricity among the group's members. This measurement follows both the group self ambitions and the local legislation guidelines. Therefore, each member has a quota from the group power generation which represents the member ownership over a given energy portion. And it is up to the member her-/himself to determine how she/he would like to exchange her/his portion.

In order to keep the group objectives of sharing the power generation and the costs to do so as a priority, without be limited by trust factors due to members nature or bad system behaviour, the proposal determines that any modification on the current quota distribution should be made in function of a token common for the whole group. Therefore, the token Sharing Electricity in Brazil (SEB)<sup>16</sup> represents the group crypto-currency and its reliability is guaranteed by the blockchain network. This method gets rid of mistrust, gives transparency for the whole group about any quota change, and allows a safety inspection with updated values for the utility and the legislative bodies.

Such as the ICO's initiatives, the tokens of the group have a particular method to be created. In the present case, every time a new power plant is added into the group power capacity, a new equivalent crypto-currency amount is created. This methodology keeps the market simple when considering new power plants and the variety of members

---

<sup>16</sup> In allusion to the term Brazilian Electricity Sector (SEB in Portuguese).

on each investment round. The token issued has a low variability on the intra-market because it is proportionally created by power units, and its purpose is to exchange energy solely. However, the electricity price is variable and relies on each power plant cost to generate that energy, that depends on different weather and fund scenarios. This makes the market unique for each member, i.e., it is up to each member's strategic vision the decision to invest in new power plants or to exchange her/his quota to get profits beyond any investment made to only cover her/his power demand.

This approach is similar to the NRGcoin (MIHAYLOV et al., 2014) in which tokens are created by raising the power generation capacity in the distribution grid, instead of spending energy on computational power to do so, as it used to happen with cryptocurrencies based on PoW methods. Curiously, new power plants will emerge only when desired by members to feed their electricity needs. So, what makes the dynamic of the internal market is the growth of members and their financial investment capacity (and interest) to fund new distributed power plants.

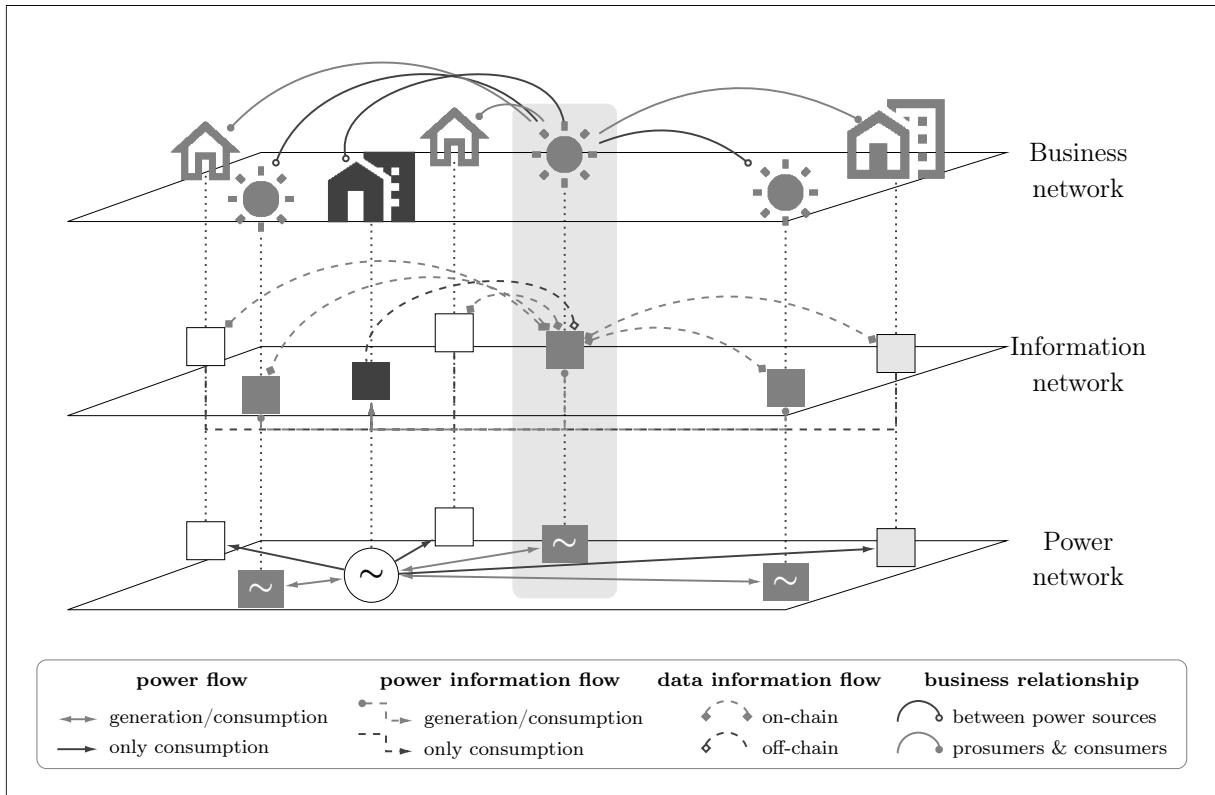
Therefore, there is no need to store money to exchange back the tokens to fiat currency, because each token represents individual money savings by the difference between the tariffs of the group and the utility. Thus, the income happens similarly to the energy credit process, i.e., each member gets her/his savings after a month-round period. Based on these, the DG can be considered an investment portfolio with periodic returns, and the token is useful to exchange quotas without directly rely on fiat money. It allows the group to determine its own valuation of the different power sources, and it gives to each member the opportunity to speculate how much her/his quota worth at any given moment, i.e., how much she/he is inclined to accept to exchange the quota.

Thus, the proposed DApp context can be detached into three main layers as shown in Figure 3. The information network layer is the key part of the development because it redefines the information flow interaction that comes from both remaining layers. So, the traditional information flow (orthogonal dashed lines) about the power grid network between consumers/prosumers and utility is unchanged, the power meter continues to register the electricity and send this information in a one-way direction. Similarly happens with the fundamental practices of the group of shareable consumption on the business network layer (curved solid lines). The new approach is the information management supported by the blockchain advantages (curved dashed lines), in what now everyone has reliable access to their counterpart members data, and an impartial communication channel to alter their quotas.

Moreover, the link between both business and power networks through the information layer is the member her-/himself because she/he is the only one with access to her/his electricity meter information and with write-permission to the blockchain network at the same time. The member as a blockchain node is responsible to input the required measure values in the information network (off-chain service) and to interact with its

peers (on-chain service). Therefore, each member is also responsible to guarantee the trustfulness of the network since off-chain services are more susceptible to fraud and errors if not properly handled by automatic methods.

Figure 3 - The operational layers of a shareable generation group supported by a DApp.



Legend: There is only one prosumer point-of-view (highlighted in vertical) because the number of curved (dashed and solid) lines arriving to/leaving from it must be replicated for every group's node.

Source: Adapted from Wang et al. (2018, fig. 1). Icons from Material Design (material.io).

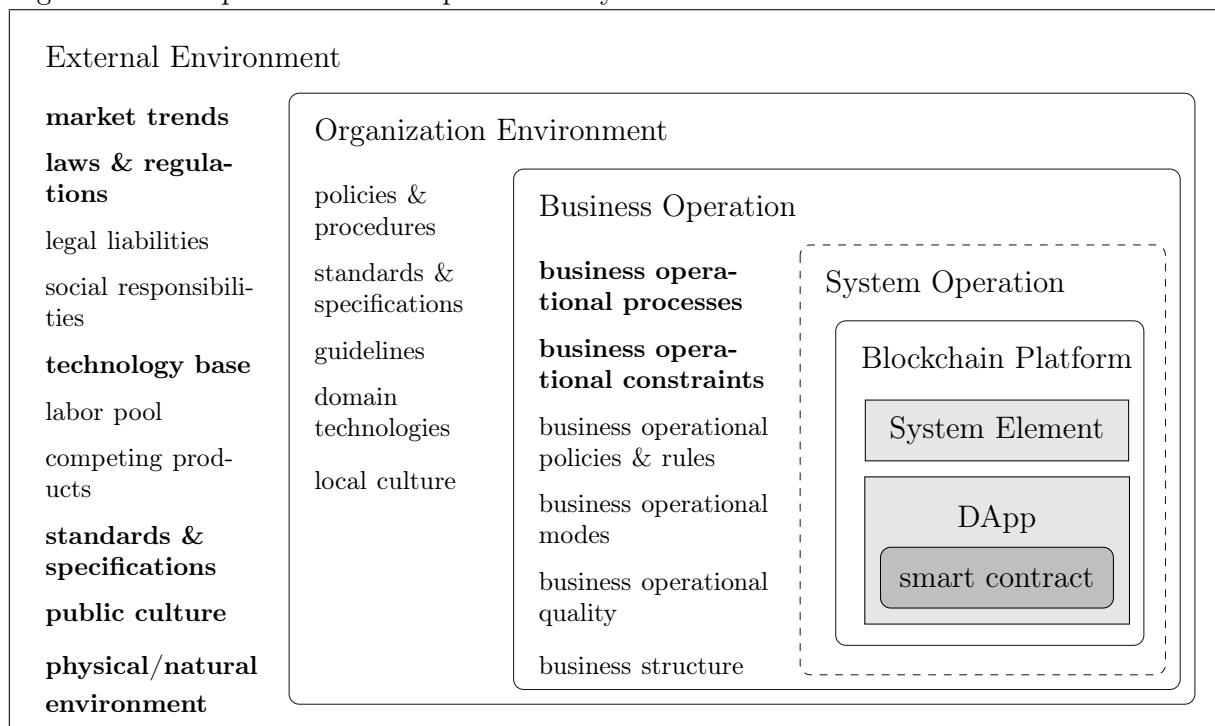
Finally, this section has enlightened that the proposed blockchain application is a small part of an entire management system to be used by the shareable consumption group to enable an intra-market for its members to negotiate their electricity fraction earnings. Let us now turn to explain the course of the DApp development. So, with the support of the ISO/IEC/IEEE International Standard of Systems and software engineering – Life cycle processes – Requirements engineering (2011), Section 3.1 describes the ordinary requirements for the DApp developed. Section 3.2 summarizes some of the blockchains available on the market and adds a comparison about them to justify the one used to develop the application. Section 3.3 presents the development course itself, detailing the application functions through pseudo-codes. Succeeding, Section 3.4 shows an example of the use of the smart contract for the TE in the test environment. Note that some limitations and constraints were encountered and described in Section 3.5. On the forthcoming section (3.6), the group market size is analyzed to contrast with the operating cost of the smart contract (Section 3.7) and the challenge to uphold the DApp.

### 3.1 Identification of the application requirements

The design of an application requests the understanding of some system statements which captures the user needs, and associated constraints and conditions. These requirements involve a set of specifications until reach the definitions of the software functions, performance, design constraints, and attributes (ISO, 2011). Figure 4 states the systematic approach of the whole process to develop a software, and highlights what the present work is up to toward the proposed DApp.

The different sets of requirement information items shown the broad range of knowledge to be considered for a fully decentralized system development. Notwithstanding, the identification of each item requires interaction and cooperation among stakeholders, mainly to define how they interact through sets accordingly with business and system specifications.

Figure 4 - Concepts for the development of a system.



Legend: The grayer boxes and the bold texts highlight the propositions considered for the blockchain application development.

Source: Adapted from ISO (2011, fig. 4).

Figure 4 presents the development scope from the top level environments, to the business level, to the “specific system-of-interest” (ISO, 2011). The first addresses the external directives the system must follow, and some of the items (bold texts) had already been discussed on previous chapters. The second refers to the intended way of doing business, and the system is mainly viewed as a black-box. It is up to the group’s guidelines and rules, and most of its items are out of the scope of the present work. The proposal

considers only what may be universal for this business category, such as power capacity size and minimal values to be exchanged. At the third stays the user's viewpoint to interact with the aforementioned scopes (ISO, 2011).

Even not aware about all specifications, the core concept of the application can be developed with no additional issues. In the future, the remaining features may improve the application presented here. For instance, the definitions about how the registering of members will be conducted and how the user interface will be designed are out of the core of the application, i.e., these are not required to securely allow the transaction between members in a doubtful environment.

Moreover, the case where the DApp relies on off-chain platforms, such as the power meter or electricity bill of the members to gather the values from generation to input in the blockchain is another exception. For both methods, the group guidelines must address what should be done to solve possible problems to access these kinds of information or if a violation is found. Although a smart contract can be created to deal with them, the subjects related to errors and penalties are not covered here.

However, for sure, the application must be accessible in any device and the user experience must encompass a wide audience age. The usability to request or to send tokens must be as easy as to send a text message, and it may be provided by the blockchain platform chosen throughout the features available for the development. Although it is an important aspect of the DApp project design, the related aspect is a hidden layer of the front-end project, which is not instigated either.

For what concerns members data, all the users must provide the same registration data available on their electricity bill at the first access to the blockchain environment of the group they belong. This constitutes sensitive data and it will be hidden from the public ledger. Indeed, this process represents the step in which a user becomes a member of a group. So, members may access those data but users not.

This information is important to identify where members reside and if they pertain to the same power utility. Although they could be in the same geographic area, this not implies being fed by the same distribution grid. It is not restricted to form a group with those customers, but a transaction between them is. Thus, each member must be classified by power utility in order to exchange tokens.

In addition, some mistakes may happen during a quota transaction, for example one could type 100 instead of 10 when writing up her/his clause. In other cases, get rid of the quota should be mandatory by legislation force. As the group may have a representative committee to act on their behalf, the application must allow corrective transactions to interfere on these issues accordingly to the group guidelines to do so. It is important to remember that blockchain ledger is an append-only database, so a new transaction must be held to update any type of information. This measure doesn't compromise privacy, neither security, but guarantees a safe interference for its members.

However, to simplify, the group representation by members hierarchy was replaced by a voting process to validate the changing of any group asset.

Lastly, the stakeholders involved in the system are: (i) consumers, (ii) prosumers, (iii) power utility, and (iv) legislator. The (i) and the (ii) are represented by the basic role of a member of a given group. They are the unique users with full interaction capability with the application, and they can interchange position. More commonly, the former can become the latter, but the opposite is not impossible to happen. The (iii) and the (iv) can read the public ledger to follow the changes in the quota, but they can only interact with few application features. For these ones, each member is only a hash (the public key) which is cross-referenced on their own private database to get full identification.

For the sake of simplicity, the proposal considers that a member joins only one group, but no restriction was imposed in the code lines. So, two members who reside in the same geographic area and who are fed by the same distribution utility may be or not part of the same group of shareable generation.

In summary, Appendix B gathers the terms that will be used throughout the publishing in addition to what has been presented so far. Note that they do not represent the variables names, although they have been used to give readability for the code. The application comprehension was designed through a UML class diagram detailed in Appendix C. There can be noticed the relationships and attributes of the *reader* (anyone with access to the public ledger), the *user* (of a blockchain) and the *member* (consumers and prosumers of a given group). Similarly, the main concepts of the blockchain technology were detached to better identify the aforementioned application features.

### 3.2 Choosing the right platform

The purpose of this section is to explore the specifications of most popular blockchain technologies available in the market. It was not an exhaustive search for particular benefits but a look for pieces of information that fill the requirements indicated previously, and then use it to compare each other, as briefly indicated at the end of the section.

#### 3.2.1 Bitcoin

As described on the beginning of the previous chapter, the Bitcoin goal was to “propose a solution to the double-spending problem using a P2P network” (NAKAMOTO, 2009, p.1). With minimal network structure, the Bitcoin follows the P2P architecture and uses the longest consensus chain as proof of whatever happens on the network. For this case, the algorithm is the PoW.

However, to accomplish with its goal without a trusted party to do so, the blockchain must be public. So, every node can verify by itself that the double-spend has not happened. Shows the transactions publicly does not compromise privacy because users are protected by their unique public key that must be kept anonymous to prevent being easily identified on the network.

The protocol also ensures that each user has access to a system with a single appending order history. Even if different versions of a block arise, nodes work on the one they receive first and monitor the other branch because nodes always “consider the longest chain to be the correct one and will keep working on extending it” (NAKAMOTO, 2009, p.3). So, after a next consensus-round, if nodes are on the wrong branch (the smaller one), they switch to the right one. Similarly happens when a node does not receive a block, its ledger will be correctly updated on the next appending block.

Another aspect to be considered is the cost of the platform. Although there is no fee to receive Bitcoins (symbolized as BTC), it is possible to adjust how much one would like to pay when spending, to encourage a faster transaction confirmation. The minimum cost for a transaction be added on a block is 0.00001 BTC/kB by transaction size. As expected, the fee value is not depended on the amount of Bitcoins transacted, “so it’s possible to send 100,000 BTC for the same fee it costs to send 1 BTC” (BITCOIN.ORG, 2019).

Furthermore, the current limits on traditional payment options are also overcome. The administrative costs to deal with fraud, and to storage and process sensitive data of customers are extinguished. This results on transactions of any amount with anyone in the world at a speed of 10 minutes on average to get a confirmation score “that indicates how hard it is to reverse them” (BITCOIN.ORG, 2019), i.e., with enough confirmations to prove a transaction as true. Moreover, the duties of an accounting transparency keep smooth with ease access to transactions on the blockchain to verify any balance requested. Another powerful feature for organizations is the multi-signature option, “which allows Bitcoins to be spent only if a subset of a group of people authorize the transaction” (BITCOIN.ORG, 2019).

The benefits for users, in general, is extended with the platform usability. The exchange can be made just displaying an address or QR code. Also, the Application Programming Interfaces (APIs) available help to filter the information needed for each application, even allowing custom options to deal with local currency payments at competitive costs (BITCOIN.ORG, 2019).

While the Bitcoin network does not provide a complete distributed system to create a DApp, it is possible to rely on its crypto-currency to establish proof of an asset transaction. For instance, alt-coins are Bitcoin protocol variations used for particular application purposes. This has introduced several other currencies to deal with different decentralised system needs. On the other hand, some options just complement the Bitcoin

features to get similar results without the need to create a new crypto-currency (WOOD, 2016).

All the aforementioned alternatives have in common the need to buy BTC to exchange any other asset, and to set a side centralized server to support these operations. Although the transaction time may be different depending on the option chosen, the price of a Bitcoin will always be volatile<sup>17</sup>.

### 3.2.2 Ethereum

The Ethereum blockchain complements the Bitcoin principles going beyond the financial concept adding the possibility to build DApps based on a unique compute resource. It has stated the transition towards the second generation of the blockchain development history, as indicated previously on Section 2.2. Accordingly with Wood (2016, p.1), the main goal of Ethereum is to provide a system “to facilitate transactions between consenting individuals” by an enforced autonomously agreement algorithm to work as a trusty impartial actor.

Ethereum is considered a transaction-based state machine, since only the final state is accepted as the valid version of the blockchain network, no matter how the genesis state was defined and how the upcoming transactions have been modifying it until the last state. Each state is composed by any computer-based information, while each transaction represents the link between two states.

However, not all state is a valid one due to bad developing-practices to handle with constraints. The alternative to solve this issue is to charge for each transaction, so developers have to take care of their DApp codes to run it cheaper and successfully, i.e., to avoid to pay for an invalid transaction “all programmable computation [...] is subject to fees” (WOOD, 2016, p.7).

The crypto-currency of the Ethereum realm is the Ether, represented by ETH. It ranges from 1 to  $10^{18}$  with incremental nominations for some of its portions, such as from Wei ( $10^0$ ) to Szabo ( $10^{12}$ ) to Finney ( $10^{15}$ ) and up to Ether ( $10^{18}$ ). Every time a block is mined, 5 Ethers are created<sup>18</sup>. Nonetheless, transaction fees are charged in units of gas that has a price defined in Ether. This means a defined cost for all over the smart contract creation, message call, account storage use and access, and any other execution

---

<sup>17</sup> For instance, 1 BTC used to cost US\$ 8.288,13 on 08/30/19 accordingly with coinmarketcap.com. How much does it cost now? How much has it been changing?

<sup>18</sup> More information about how this works is available at: [ethereum.org/learn/#proof-of-work-and-mining](https://ethereum.org/learn/#proof-of-work-and-mining).

operations on the virtual machine (WOOD, 2016, Appendix G).

The crypto market dynamics are guided by how much each miner is up to exchange her/his computing resource for a transaction. This determines a gas price offering table based on Ether. A user can also specify how much would like to pay for a gas price and so miners decide what it is better for them. In the end, miners compete between them to gain Ethers doing a trade-off between their gas price and transaction time capability. And users may get faster transaction time accepting higher gas prices.

Note that “gas does not exist outside of the execution of a transaction” (WOOD, 2016, p.7), so all Ether used to buy gas for an invalid transaction is wasted. But nothing happens if someone does not have enough Ether to pay for a transaction. It is possible because Ether payment and withdraw are handled on different steps. Firstly, no contract cost is paid by a user until a transaction is validated on the initial tests of right code structure, signature, gas limits and so on. Lastly, the Ether used to buy gas for the corresponding transaction is given to the beneficiary miner only after the transaction turns into a new state on the blockchain.

Besides the fees to the contract computation and invoke/creation, a charge also applies for an increase in the usage of the system storage. It aims to reduce the use of the storage because it directly influences on “a larger state database on all nodes” (WOOD, 2016, p.10). Indeed, this measure is so important that a refund is given for any operation that gets rid of an entry in the storage space.

Although Ethereum also uses PoW for the mining process<sup>19</sup> like as Bitcoin does, there is a variation on the protocol used to reach consensus. Ethereum uses a simplified version of the GHOST protocol (WOOD, 2016) to communicate miners about a new state on the network. This has improved the average block time to 12 seconds.

Another difference between both blockchains is that Ethereum transactions also consider smart contract output operations (or exceptions) besides any information about creation and transfer of Ethers. Thus, the block structure is formed by “a mapping between addresses and account states”. The former “is cryptographically dependent on [the latter] and as such its hash can be used as a secure identity for the entire system state” (WOOD, 2016, p.3). While addresses are kept on the network chain, the data structure remains in a underlying database.

To allow the DApp performance, the Ethereum Virtual Machine (EVM) is responsible to execute specific instructions to change the system state. However, the total amount of computation is limited by gas. The EVM is a simple architecture of a stack machine with an independent non volatile storage. Every program code keeps on a sepa-

---

<sup>19</sup> Actually, it is expected a move towards the PoS. More information about it at: [docs.ethhub.io/ethereum-roadmap/ethereum-2.0/proof-of-stake](https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/proof-of-stake).

rate virtual ROM and is accessed only through a particular instruction. At the end of an execution process, all the output is recorded on the ledger as part of a new state.

### 3.2.3 Hyperledger

Different from the others, the Hyperledger is not a unique blockchain technology, but a family of DLTs designed to fit diverse business requirements and allow cross-industry solutions. It is an open source collaborative effort under the Linux Foundation with leaders in finance, banking, Internet of Things (IoT), supply chains, manufacturing and technology (HYPERLEDGER, 2018b).

Nonetheless, all the five frameworks<sup>20</sup> available by the Hyperledger make use of a modular architectural design in order to “encourage the re-use of common building blocks [and to] enable rapid innovation of the DLT [...] and the interfaces between them” (HYPERLEDGER, 2018a, p.13). Consequently, it has the benefits of extensibility and flexibility, allowing independent modification of any component in an interoperable way, offering highly secure solutions through rich and easy-to-use APIs (HYPERLEDGER, 2018b; HYPERLEDGER, 2018a).

Aware that business blockchain networks operate under a partial trust environment, and the requirements may represent a potentially unique optimization point for the technology (HYPERLEDGER, 2018a), the resulting system design relies under the classification of permissioned blockchain network but the range between public and private definition varies by each characteristic of a specific framework. Similarly, the whole family has a particular consensus method that not include the standard PoW with anonymous miners, either the use of a native token nor a crypto-currency (HYPERLEDGER, 2018b).

Anyhow, four frameworks use a voting-based approach to consensus. Hyperledger Fabric, Hyperledger Iroha and Hyperledger Indy by means of the algorithms Apache Kafka, Sumeragi and Redundant Byzantine Fault Tolerance (RBFT), which are nothing than variations of PBFT method. And the Hyperledger Burrow uses PoS through Tendermint consensus engine<sup>21</sup>. On the other hand, Hyperledger Sawtooth uses a lottery-based approach, the PoET (HYPERLEDGER, 2018b).

The Hyperledger business modular approach is quite similar to what was shown previously at Figure 2 (Section 2.3.1). Besides the consensus layer purpose (Layer 0), the remaining Hyperledger layers dive into business blockchain components over the gen-

---

<sup>20</sup> Recently, it was announced a new framework, the Besu, that can also run on Ethereum. More information at: [hyperledger.org/projects/besu](http://hyperledger.org/projects/besu).

<sup>21</sup> More information in the project’s repository: [github.com/hyperledger/burrow](https://github.com/hyperledger/burrow).

eralized layers already presented. For instance, Hyperledger's smart contract layer corresponds to the Layer 2a, but the provided data-storage module and identity services module can be correlated with the whole Layer 2. Independently of where they stand (on- or off-chain) their purpose are to allow the better business fit by other modules.

In addition, there is a policy service module responsible for management of various system policies. It is a significant requirement to rule the application behaviour over all other modules, in this way it can be consistent with the Layer 5 (DApps). The APIs module concerns usability such as the Layer 4. Another module is the interoperation itself (Layer 3), which supports interoperability between different blockchain instances. The remaining modules about communication and crypto abstraction can be correlated with the same layer. The former aims to allow “peer-to-peer message transport between the nodes that participate in a shared ledger instance”. Whilst the latter aims to support “different crypto algorithms or modules to be swapped out without affecting other modules” (HYPERLEDGER, 2018b, p.3).

What concerns with financial subjects, some operations costs rely on the gas price incurred from the integration with Ethereum but others are like the pricing models for cloud computing, as the case with Hyperledger Fabric on IBM Blockchain Platform<sup>22</sup>. In essence, the pricing models follow the business of the blockchain system principles which applies time or token-based techniques to ensure limited resource consume of the network (HYPERLEDGER, 2018a).

### 3.2.4 Neo

The Neo blockchain defines itself as a “distributed network for the Smart Economy” (NEO, 2018). Based on the principle of digital identity, i.e., real identity information stamped in electronic form, Neo allows a true physical asset ownership by means of a digital asset that, in the near future, can even replace the Online Certificate Status Protocol (OCSP)<sup>23</sup> to manage and record the X.509 Certificate Revocation List (CRL) (NEO, 2018).

The Neo blockchain digital assets have two forms of operation. One known as global asset refers to the public system space, which involves the native assets NEO and GAS. The other one refers to the contract assets, that may follow some standards to keep

---

<sup>22</sup> The pricing of IBM Blockchain Platform for IBM Cloud is available at: [cloud.ibm.com/docs/services/blockchain?topic=blockchain-ibp-saas-pricing&locale=en](http://cloud.ibm.com/docs/services/blockchain?topic=blockchain-ibp-saas-pricing&locale=en).

<sup>23</sup> A protocol that defines the type of data exchanged between client and server under a SSL communication. More information at: [ibm.com/support/knowledgecenter/en/SSLTBW\\_2.2.0/com.ibm.zos.v2r2.gsk100/cert\\_revocation.htm](http://ibm.com/support/knowledgecenter/en/SSLTBW_2.2.0/com.ibm.zos.v2r2.gsk100/cert_revocation.htm).

compatibility over the whole system if its intention is to create a side crypto-currency, and may also have a private storage, an area created by a smart contract with restrictions to interface with.

Likewise most of blockchain technologies, Neo uses the native tokens to govern transactions. The token NEO by itself represents the right to manage the network, such as voting for bookkeeping and involvement on network parameter changes. Its value ranges from 1 to 100 million with integer steps, without subdivisions. Its total amount was created in the genesis block and it was half divided between the supporters of NEO during the ICO and the Neo Council. The latter manages the tokens to support “Neo’s long-term development, operation and maintenance, and ecosystem”. In that case, tokens “will not enter [into] the exchanges [processes]” (NEO, 2018). Therefore, who holds NEO is part of the network owners and has right to manage the network by voting procedures.

The other token is the fuel to control the Neo network resources. It is called NeoGas, abbreviated as GAS, and has the same maximum total limit of 100 million, although its minimum unit is  $0.00000001$  ( $10^{-8}$ ). It is created with each new block but has a controlled increase by a annual decay algorithm rate, and its distribution is made proportionally to each NEO a node has. Although GAS is used to charge transactions and smart contracts, some of the operations have no fee<sup>24</sup>. In addition, some NeoIDs have priority when a large amount of transactions happen, while others may get this benefit by paying additional GAS.

This option is valuable because the Neo consensus mechanism has an interval between each block of about 15 seconds. Furthermore, the distributed storage protocol utilizes the same technology used by IP management over the whole Internet, the Distributed Hash Table (DHT) technology (NEO, 2018).

The blockchain also supports large files. They are “divided into fixed-size data blocks that are distributed and stored in many different nodes” (NEO, 2018). This feature might require a trade-off between redundancy and reliability, that aims to use token incentives and backbone nodes to solve it, so users may set their requirements for a given file. In the end, a file priority level will be proportional to its assigned cost of storage and access.

In order to allow the execution of smart contracts, the Neo Virtual Machine (NeoVM) is a lightweight virtual machine “which can implement arbitrary execution logic and ensure consistent execution results of any node in a distributed network, providing strong support for decentralized applications” (NEO, 2018). Indeed, no new development language is required to develop Neo smart contracts, since codes written in Java or C#, for instance, can be compiled into a unified NeoVM instruction.

---

<sup>24</sup> At least based on what have been defined so far by Neo at: [docs.neo.org/docs/en-us/sc/fees.html](https://docs.neo.org/docs/en-us/sc/fees.html).

Likewise the aforementioned blockchains, Neo also has APIs to track native and side crypto-currencies over smart contracts and transactions, besides methods to query the blockchain and manipulate the persistent storage through the Smart Contract Execution Engine presented at the NeoVM.

### 3.2.5 Corda

“R3 is an enterprise blockchain software firm [that was born] to leverage blockchain technology to solve real business problems in both complex and highly-regulated markets” (R3, 2019). It was initially supported by world’s leading financial institutions but now it has “more than 300 participants across multiple industries” (R3, 2019).

Its project has started with an open source blockchain platform called Corda<sup>25</sup>, for free download, which “enables businesses to transact directly and in strict privacy using smart contracts, reducing transaction and record-keeping costs, and streamlining business operations” (R3, 2019). Nowadays, it also has one more blockchain distribution, the Corda Enterprise, “a commercial version of Corda which offers features and services fine-tuned for modern-day businesses” (R3, 2019).

Although its platform has a similar approach to the Hyperledger’s one, i.e., targets value proposition through business applications, Corda allows for every network participant to interchangeably transact between them aside their business nature and asset that could limit it. This is different from what is presented on other blockchains, where each enterprise creates its personal environment with a singular asset that cannot be directly exchanged outside its space. For R3, the methodology based on silos is the opposite of what an enterprise blockchain is looking for, so they made some adjustments to the network architecture which resembles to permissioned private blockchains.

R3 identified that most of the drawbacks from general consensus methods over the whole blockchain network can be avoided by using a simplified consensus mechanism based on mutual agreement with a real digital identity. Through X.509 certificates, Corda associates a unique legal name with at most one public key and IP address. Therefore, any transaction a participant may do is unambiguously binding to her/his legal entity, doesn’t matter if it is a person nor a company. At the end, this premise offers valued services from multiple providers upon their relational consensus at a given time enforced by a smart contract (BROWN, 2018).

Furthermore, the smart contract deployments follow the already cited structure of other platforms, however its applications are known as CorDapps and runs on Java Virtual

---

<sup>25</sup> It is available in the project repository: [github.com/corda/corda](https://github.com/corda/corda).

Machine (JVM). Another difference is that Corda does not create a block of concatenated transactions to define a new state on the ledger, because each transaction is a new block and a new state indeed. The R3 block fundamental is a proof-of-record of a “specific instance of a specific agreement, which may be thought of as representing a real-world contract or section of [it]”. (BROWN, 2018, p.8)

Moreover, the Corda consensus goes beyond the rules defined on smart contract as part of CorDapps. Through unique and time-stamping services, transactions are temporally ordered and conflicts are eliminated later on by a dedicated framework that coordinates complex multi-step protocols without a central controller (BROWN, 2018).

Therefore, the shared ledger can be seen as just a distributed database that keeps the network data reliable but each block content is encrypted and only accessed by the counter-parties that have signed it. “Thus, any given actor in a Corda system sees only a subset of the overall data managed by the system” (BROWN, 2018, p.10), i.e., even being “allow[ed] arbitrary combinations of actors to participate in the consensus process for any given piece of data”, the block content “is intended to be shared only with those who have a legitimate reason to see it” (BROWN, 2018, p.8).

Finally, its cost structure also follows the cloud computing model to deal with storage<sup>26</sup> and with transaction fees<sup>27</sup> besides the annual participation (membership) fee to be a legal entity of the Corda Network<sup>28</sup>.

### 3.2.6 Libra

Announced on the current year, the Libra blockchain is the result of a consortium led by Facebook with not-for-profit organizations and a Swiss foundation to create a global currency to support worldwide people with a powerful financial infrastructure (BINANCE RESEARCH, 2019). This blockchain aims to advance financial inclusion through an initiative approach that “can deliver a giant leap forward toward a lower-cost, more accessible, more connected global financial system” by means of collaboration with regulators and experts “to ensure a sustainable, secure and trusted framework” for an innovative system (LIBRA, 2019).

Likewise Bitcoin, its main crypto-currency has the same name of the blockchain.

---

<sup>26</sup> The pricing scheme is available at: [aws.amazon.com/marketplace/pp/R3-Corda/B07MXHSR6P/#pdp-pricing](https://aws.amazon.com/marketplace/pp/R3-Corda/B07MXHSR6P/#pdp-pricing).

<sup>27</sup> The costs to use the service is presented at: [corda.network/participation/membership-tiers.html#2transaction-fees](https://corda.network/participation/membership-tiers.html#2transaction-fees).

<sup>28</sup> The membership fee is at: [corda.network/participation/membership-tiers.html#1annual-participation-membership-fee](https://corda.network/participation/membership-tiers.html#1annual-participation-membership-fee).

However, Libra is only a payment token backed on four fiat currencies: the United States Dollar (USD), the Euro (EUR), the Japanese Yen (JPY), and the Great Britain Pound (GBP) (BINANCE RESEARCH, 2019). Moreover, like other blockchains with two tokens, the Libra blockchain also has the Libra Investment Token (LIT) to allow participation in the governance of the network. But it works separately to the first one, being similar to an ICO process to enable the success of the network (BINANCE RESEARCH, 2019).

The blockchain architecture uses a PoS consensus with a custom type of PBFT algorithm, called LibraBFT, which “is a variant of the HotStuff framework that was released in 2018 by Maofan Yin and Dahlia Malkhi of VMware Research. This framework may allow for future interoperability with other chains” (BINANCE RESEARCH, 2019, sec.2). It currently fits as permissioned blockchain but aims to transit towards a permissionless one when ready to operate (BINANCE RESEARCH, 2019).

Another particular feature is the programming language. Similarly to Ethereum, Libra blockchain has its own commands to develop smart contracts, named Move. This language encompasses some components beyond the bytecode language write format, such as a suitable programming model for the blockchain execution, a “module system for implementing libraries with both strong data abstraction and access control”, and a “virtual machine consisting of a serializer/deserializer, bytecode verifier, and bytecode interpreter” (BLACKSHEAR et al., 2019, p.19).

Although the Move virtual machine “executes a block of transactions from a global state and produce a transaction effect representing modifications to the global state” such as other blockchains VMs do, the resulting effect are checked before turn into a new block state. This behaviour to handle the effects separately “allows the VM to implement transactional semantics in the case of execution failures” (BLACKSHEAR et al., 2019, p.18). In addition, transactions still are executed sequentially “but the Move language has been designed to support parallel execution” (BLACKSHEAR et al., 2019, p.19). And to avoid overhead of Move programs execution, “each bytecode instruction [also] has an associated gas unit cost, and any transaction to be executed must include a gas unit budget [...] similar to EVM” (BLACKSHEAR et al., 2019, p.15).

### 3.2.7 Comparison between the blockchains

For the comparison analysis of the most popular blockchain technologies presented on previous subsections it was considered only those with full capability to develop DApps and with relative importance for the present work. Therefore, neither Bitcoin, Corda, and Libra blockchains were took into account.

Although the proposed application can be adjusted to be an alt-coin of the Bitcoin, this is not enough to manage the group’s power asset with confidence because this infor-

mation will stay on a third-party storage space, which implies on low audit transparency. In addition, the Bitcoin price volatility may harm the purpose to exchange and to share the benefits of a distributed power generation and management.

The Corda's already established concept of interoperable blockchain applications among different businesses bears with the desire of other blockchains, as previously discussed at Section 2.3.1. So, it is expected an upcoming integration between platforms on the near future. While it is good for different businesses to share a common environment to exchange assets, the present application does not require such kind of feature so far.

Similarly applies for Libra which focus is to embrace affordable social payment between different fiat currency systems. For sure it will impact the world for any kind of financial subjects due to the presence of Facebook applications on the daily life of people but its features does not attend the requirements for the DApp under consideration. For this reason, the comparison relies only between Ethereum, Hyperledger, and Neo, as summarized on Table 4.

Even though Table 4 presents a considered number of important differences between the blockchains, the greater characteristics for the final decision are those related to the Blockchain Application label. Firstly, the permission for a private storage space is a crucial aspect, since some members decisions with sensitive information may be held on the blockchain and it must be kept private from the outside group-world. Consequently, the group can define its own rules to handle with these pieces of data and create its own consensus method to approve or discard any change beyond the blockchain consensus. For instance, a ballot process or a board of governors to act on the group behalf using the multi-signature feature to decide upon a request.

Secondly, the implicit costs to allow advantageous use of the smart contract are other important points to be considered on. Although read and write functions are usually charged on side coins, they can be indirectly correlated to a common fiat-currency by means of how much the respective main coin is evaluated on the finance market. For instance, Hyperledger does not have native crypto-currencies, neither a pricing pattern, and the read/write operations differ by framework types. To get a cost summary of all Hyperledger frameworks without a specific detailed business case is nonviable. On the other hand, Neo differs from Ethereum in how *gas* are created. At the latter, Gas is priced every time a function is processed with an almost fixed conversion rate between ETH and Gas. While it may sound good, the amount of ETH decreases over time due to operation payments. And then, it will be required to buy more crypto-currencies to pay for future operations, which may compromise the DApp due to the ETH price volatility. By contrast, the former creates GAS by a fixed rate upon how much NEO one has. Once NEO is acquired, GAS will always be created until its maximum value allowed. This behaviour gives control to the group to make a trade-off between expenses with read/write operations and how much NEO coins are required to support the quantity

of the related operations.

Table 4 - Comparison of some blockchains.

	Ethereum	Hyperledger	Neo
<b>General Specification</b>			
blockchain scope	permissionless	variations of permissioned	permissioned public
main coin	ETH	-	NEO
main coin range	1 to $10^{18}$	-	$1 \text{ to } 10^8$
side coin	Gas	-	GAS
side coin range	varies per transaction	-	$10^{-8} \text{ to } 10^8$
side coin creation	by type of operations	-	almost constant
allow alt-coin?	yes	yes	yes
allow multi-signature?	yes	depends	yes
<b>Consensus Protocol</b>			
algorithm method	PoW	PoW <sup>a</sup> , PoS, PoET, PoA, PBFT-like	DBFT
block average time	12s	depends	15s
allow faster block time?	yes	depends	yes
allow being part of the decision-maker network?	maybe <sup>b</sup>	at some frameworks	yes
<b>Blockchain Application</b>			
usability?	yes	yes	yes
read cost <sup>c</sup>	complex	complex	complex
write cost <sup>c</sup>	complex	complex	complex
allow private space?	yes (limited)	yes	yes
charge for private space?	yes	yes	yes
APIs?	yes	yes	yes
online fast test environment?	yes <sup>d</sup>	no	yes <sup>e</sup>
integration with off-chain platforms?	yes	yes	yes

Legend: <sup>a</sup> Due to the frameworks that uses Ethereum.

<sup>b</sup> When the expected change to the PoS happens.

<sup>c</sup> Since each blockchain has a particular business model to price the use of the platform resources, the read and write costs are complex for the comparison.

<sup>d</sup> Available at: [remix.ethereum.org](http://remix.ethereum.org).

<sup>e</sup> Available by Coelho, Coelho e Cardoso (2018).

Source: The author, 2019.

Another important cost is the one inherent from the use of private space. Neo still not charge for its use but have already signaled a transition to consider some fees in the future, as previously stated in Section 3.2.4. Hyperledger follows the same principle of

cloud resource control. And Ethereum does not work well with large data size, since it financially incentivizes the removal of data from its storage space.

Thirdly, the usability factor looks not to be a blockchain problem because all of them offer a complete API documentation about code standards, good practices and compatibility to assure full integration with particular applications. Fourthly, the integration with off-chain platforms is another key requirement. This may be seen as part of the API indeed, but a bit more specific and careful. So, for example, money transfers agreed on-chain and gathering data from power meters still depends on trust factors in the user or in an intermediary application, that could be fairly auditable in the future when automatically integrated with the blockchain.

Lastly, an online fast test environment to support developers to create and to share their smart contract codes is a must. This is rarely considered on the beginning of a blockchain project but decisive to rocket any application coding curve learning. Although Hyperledger offers options to test its frameworks' smart contracts, it lacks an online environment to do so. By contrast, Ethereum has been constantly upgrading its test environment to look more friendly. And Neo's option allows developers to write smart contracts in any Neo supported language, to save local copies of them, and to chat with other users.

Besides the Blockchain Application discussion, the aspects under the Consensus Protocol label are about minor importance because all blockchains offer what the proposed application requires. Even with the performance of any Hyperledger framework being really tough to define due to its latency measurement being dependent on the level a finality must have to satisfy a particular business case and not on overall network performance.

Differently from the financial subject that needs faster transactions, the electricity accounting is a month-based period. Likewise, the decisions for quota changes does not have an intense natural behaviour, so a fast time to validate a power transaction is not an issue. Indeed, there are relevant expectations for real-time electricity pricing in future smart grids, as presented on the literature highlighted on Introduction, but these futuristic wishings have a long way to go.

In view of all that has been mentioned so far, the Neo blockchain was chosen to develop the smart contract proposed due to how its financial methodology is defined, the affordable permission for private space, and a valuable easy-to-use test environment with NeoCompiler Eco.

### 3.3 The DApp specifications

Returning briefly to the purpose of the DApp, the members from a group of shareable consumption may use the blockchain service to safely exchange their power assets,

and give transparency about their transactions. For simplicity and elucidation only, the considered application is composed just by one smart contract, that is responsible to define the group-centered blockchain environment apart from the main blockchain network, and to dictate the commands to achieve the group purpose. The Microgrid Transactive Energy smart contract (MTEsm) was designed to this end, taking into consideration the importance to protect members' data from public access and to guarantee the transactions between them. Therefore, the following paragraphs outline the concepts of the DApp creation.

As initially observed in Section 3.1, the DApp does not cover all aspects of the management system. However its core agents, objects and relationships were identified and are detailed in the UML diagram in Appendix C. The related representation have supported the development of the following functions and operations structure. In addition, in Appendix C is possible to get a broad view of the MTEsm integration with the main blockchain network, its services, and off-chain interfaces. It also has a member-centered approach to better handle with the system management design, which can be observed throughout the subsequent parts of this section, that is completely platform agnostic, until the lines of code transcribed in Appendix E, that is dependent on the Neo specifications.

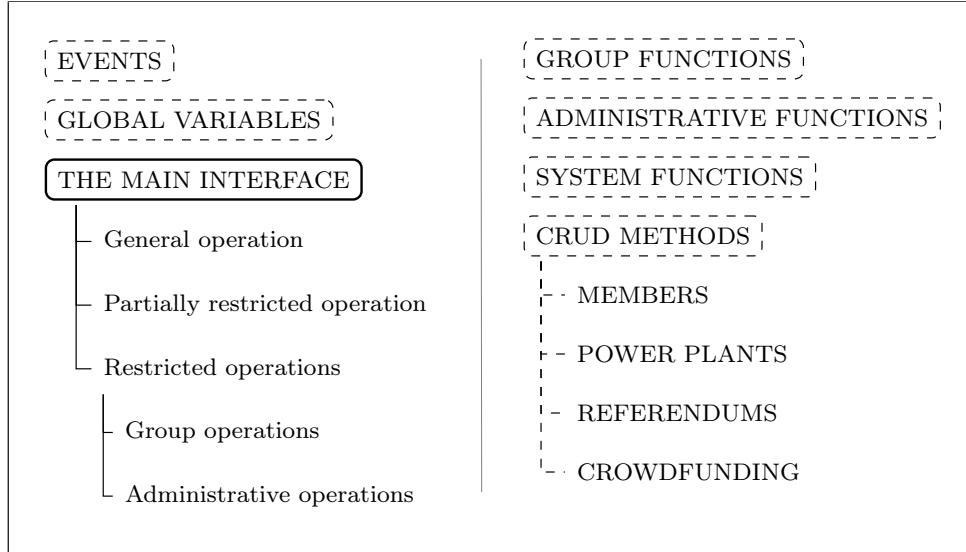
Therefore, despite Figure 5 presents an overview of the MTEsm code structure, the smart contract operations follow the definitions in the UML diagram. Then the connection between both concepts can be better understood through an organization of the functions into three sets, namely:

- General** It is the operation type that provides only one function for users to request to join the group.
- Partially restricted** It is an operation that also provides only one function to access the group data base but with restrictions based on what information is requested. So members and users have different access by each statement request, which can be about the group power capacity, the group number of members, a given power plant registration data, or a member dataset.
- Restricted** It encompasses the operations that provide the main functions for members to vote on a process, to bid on a power plant crowdfunding, or to transact between them. There are also some functions, referred as administrative ones, that complement the processes that happen off-chain. Moreover, neither of these kind of functions are restricted by members hierarchy but any other user interaction will not be allowed.

Beyond the code structure, there are four other specifications that were considered throughout the DApp's functions development. One concerns to financial subjects. So a

lot of decisions were carefully taken before reach the final desired result in order to shrink the amount of GAS consumed. This methodology is based on the suggestions provided by Jing (2018) and on the Neo computing expenses reference (CELIA, 2019b; CELIA; QIAN, 2019).

Figure 5 - The structure of the MTEsm code.



Legend: Users may interact with the smart contract through the operations available under the main interface (solid box), while the remaining functions (dashed boxes) support all the operations needs.

Source: The author, 2019.

The other concerns to the NeoVM limitations to compile the full C# library. For instance, the supported C# integral types are converted to `BigInteger` type in the NeoVM. So, initially, the MTEsm was designed with only the variables types allowed by the compiler. However, following available examples, the variables were redefined with more details accordingly with its expected values. This gave better code readability and attention to value units, even aware that in the end all integer-like variables will be converted to `BigInteger`.

The third consideration is about the IDs managed by the MTEsm. All of them follow the Base58 encoding scheme, introduced with Bitcoin for its system addresses representation in a human-friendly format. Basically, the big numbers generated at cryptography steps are converted to the Base58 alphanumeric format. Walker (2020) shows a good introductory explanation about the encoding/decoding and further details can be found on several blockchains documentations. In particular, the MTEsm has a similar but simpler function than Neo to generate its identifications, since only the encoding step is required. The recognition of this feature was important because NeoVM limits the byte array size up to 32 bytes, and to perform the checksum step used by Neo to keep credibility between encoding and decoding is not possible by a smart contract.

Although several efforts were made to keep all of the IDs following the same creation

pattern, the Neo SDK uses the **SHA256** cryptographic hash function to generate an address, which already outputs a byte array with 32 bytes. As it is needed one more byte to define the positive value on the conversion to **BigInteger**, and even some more bytes to validate the encoding/decoding steps, the solution was to use another hashing algorithm. Then, the IDs created by the MTEsm use the **RIPEMD160**, which is 20 bytes long, and they are directly converted to the Base58 alphanumeric text from the hexadecimal format.

However, the array length defines the output encoding method, and the prefixes used by MTEsm follows a different pattern from the one used by Neo, and by Bitcoin too. Therefore, the smart contract creates three kinds of IDs: (i) for referendums, with prefix R; (ii) for power plants registers, with prefix P; and (iii) for bids, with prefix B. Whilst (i) and (ii) are displayed on the output to be later used by other operations, the (iii) is never shown, since its creation depends only on the PP and member IDs, and is used for internal operations solely. Thus, along with the user address provided by the Neo system, that is also used for the member identification, the MTEsm deals with four IDs in total. An overview about them is presented in Table 5.

Table 5 - The differences between the 4 kinds of IDs handled by the MTEsm.

	prefixes			
	A...	P...	R...	B...
used to identify	members	power-plants	referendum processes	bids
created at/when	Neo register	power-plants are approved	referendums start	members bid
hash algorithm	SHA256	RIPEMD160	RIPEMD160	RIPEMD160
decode/encode check?	yes	no	no	no
compliant with Neo IDs?	yes	no	no	no
pair of cryptographic keys?	yes	no	no	no
prefix (in hex)	0x17	0x53	0x5A	0x27
validation method	CheckWitness (Neo API)	search in MTEsm storage	search in MTEsm storage	search in MTEsm storage

Source: The author, 2019.

The last specification is the time stamp format used, an important variable to deal on a distributed environment. Neo system uses Unix time stamp to synchronize its operations. It represents the date and hour running time in seconds since January 1st, 1970 at Coordinated Universal Time (UTC) (WIKIPEDIA, 2019). Therefore, all constants and variables of time were defined in this format. Even the time frames considered are multiple of seconds, for instance a 30 days period has a constant value of 259,200 seconds. On the other hand, additional care must be made when passing input variables different from this format because no time conversion was implemented, and it must be completely handled off-chain.

After understanding the basis of the development process, let's moving on to the operations performance. At a glance, the following pseudo-codes catch all cases when

faced with any kind of input, from how it is processed, to how the output is generated. These descriptions are useful to go along with the evaluation of the smart contract invoke operations, such as the example presented in Section 3.4.

### 3.3.1 The description of the operations

The objective of this section is to give an overview of the performance of each operation available through the MTEsm main interface. Without delving into the specifications of a programming language, the presented flowcharts catch the smart contract course when an operation request happens. Beyond the usual relationships between inputs and outputs, the following figures also highlight the common processes to handle with the persistent storage, off-chain operations, and displayed messages.

First of all, let's start with an evident statement: the functions run as consequence of a user interaction with the MTEsm. However, the chosen operation, along with its parameters, has to be correctly invoked as a transaction with the smart contract address to allow the function execution. Independently of the situation, it is always expected a valid response status from the blockchain VM since a failure response status means waste of money (GAS for the Neo case) and no information about what got wrong. The MTEsm tries to instruct the user about each operation do's and don'ts to reduce those issues. An explanation to avoid it is in Appendix F, and the description of each of the considered constraints is in Appendix D.

Therefore, there are two kinds of valid outputs available. One comes from the return argument, i.e., the final step of a given function's execution. It may represent any variable type and its meaning depends on how the function call and response are interpreted. For instance, a Boolean value may be either a function status of completion, or an indication of success/failure of a procedure. The other kind of output acts as a complementary “printing” argument with messages about what has been performed. This is known as *notification* and was also used to inform about identified exceptions.

Another aspect to be considered on is the MTEsm persistent space, that only starts to storage values after the first invoke for admission succeed. So, meanwhile the MTEsm private space is not fulfilled with information, the available reading operations will return empty answers, even though notifications may be displayed. This behaviour is not so easy to catch in the following figures since this kind of detail was avoided. Anyway, it is important to keep it in mind when comparing the flowcharts with the code presented in Appendix E, and because these relevant operations still have a cost.

The last note stands for the off-chain procedure. Since the whole process of some operations may consider a break to wait for some time, it was preferable to represent the off-chain step as a link between two on-chain operations. Consequently, the figures

show a continuous flow of the operation, although two invokes must happen to complete the process. The representation made with dashed rounded boxes highlights the ending of the previous operation and the beginning of the next one (of an *administrative* type). Indeed, it can be also noted that a new decision block with constraints is evaluated after such kind of boxes, which indicates another invoke operation.

Before digging into the pseudo-codes, the items below complements the classification shown in Figure 5 with the description of each operation available at the main interface:

**ADMISSION** This process is the entry way to the group environment. Any user can (*Figure 7*) make a transaction with the MTEsm address through the option *admission* to request to join the group. After a waiting period already defined in the smart contract (was used 30 days), any member may resume the process to get the request result. If approved, the user is identified inside the group, with the dataset of both profile and register, and can make use of the other operations belonging to the private environment.

**SUMMARY** It is the partially restricted operation. Depending on who has invoked (*Figures 8 to 9*) this option (user or member) different outputs are displayed. In short, a user is forbidden to access any member's information, such as to check an individual balance of tokens and quota. However, no restriction applies to visualize power plants and referendums statuses. Likewise to get a brief overview of the group with the most updated values of the total number of members and power capacity all at once.

**VOTE** Every time a referendum-like process starts, there is a voting period. (*Figure 10*) It supports a fair and transparent group decision, for instance, to approve a user request to join the group or a new power plant funding. Although the countdown happens off-chain, members should state on-chain her/his placement during the time frame of 30 days from the respective start of the timestamp process. Moreover, each member has the same power vote, i.e., 1 member is equal to 1 vote. To comply with the financial policy algorithm stated before, only positive answers are counted but both answers are displayed. However, when evaluating the result, absence votes are weighted as negative answers.

**BID** When members are notified about a crowdfunding, they can bid to get (*Figure 11*) a portion of its new power plant capacity. Important restrictions apply to get a successful transaction, such as the member must belong to the same coverage area of where the new DG will reside. The whole process also updates the number of contributions made and keeps record of each

one, so in case the crowdfunding fail and a cancellation have to be made, some information persists.

**CHANGE** Any modification on the information already established on the group's (*Figures 12 to 15*) private space is considered an update. Moreover, there are only two options that can be changed without a group agreement (i.e. without wait for a entire referendum process), that are the update of the member profile data, and the member bid on a crowdfunding, which depends on the decision of the member her-/himself only. In the case of a member deletion, the MTEsm will redistribute the quotas for the remaining members. And for a power plant elimination, no big changes have to be made on members' register since their quotas is a percentage rate of the group power capacity, not of the single value of a power plant capacity.

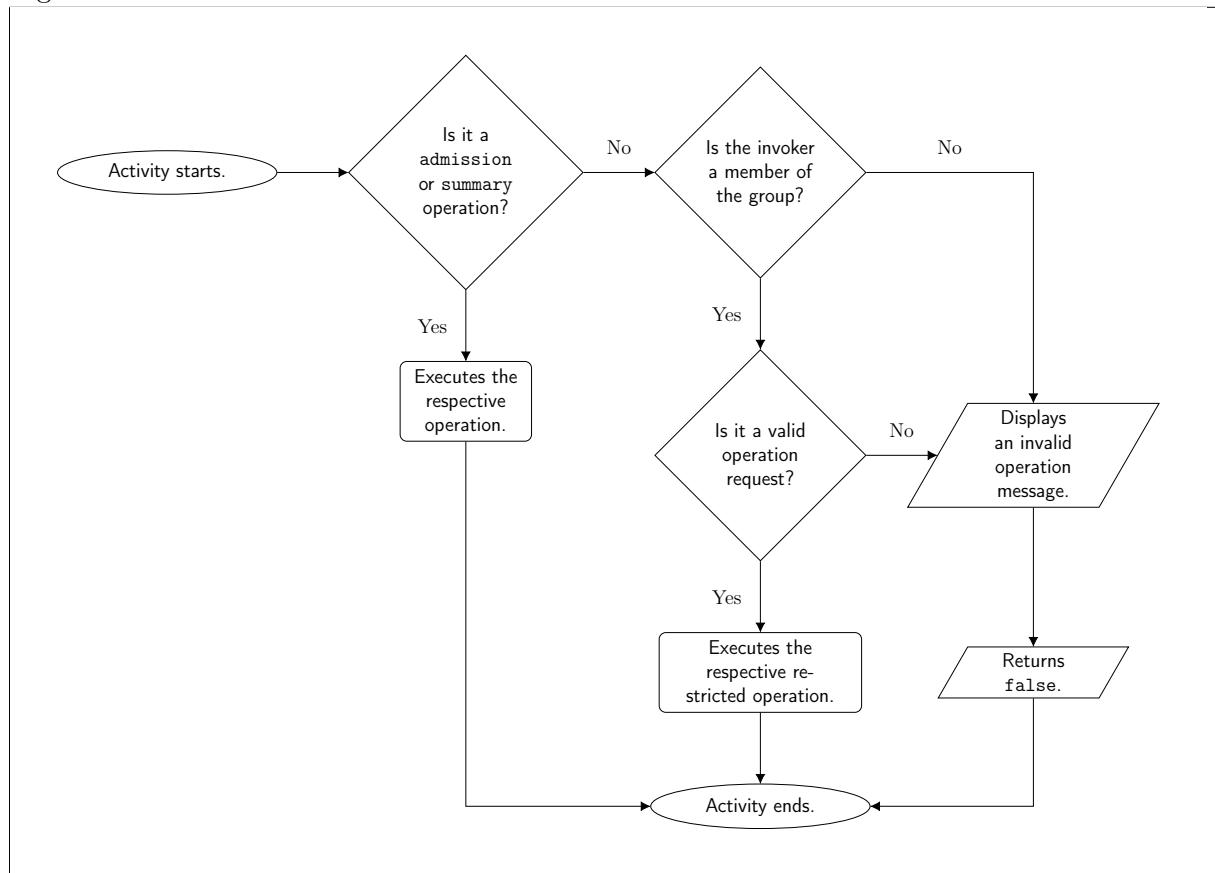
**POWER-UP** This function aims to provide the tools to manage any power plant registering data. Any member can request to add a new power plant, however, until a power generation start, a referendum process must succeed firstly. This process is different because, beyond the traditional waiting period of 30 days of a ballot, there are two additional waiting times, one for the crowdfunding (pre-defined to 60 days), and other for the power plant construction until its gets ready to operate (pre-defined to 30 days too). For the crowdfunding case, the MTEsm registers how much each member is up to pay to fund a given amount of power, but the bids coordination and payment have to be made in off-chain platforms. If funding succeed, the power plant construction period starts. The interaction with real-time planning is subtle and this variable can be updated before its deadline. This waiting time is required to avoid unbalanced distribution of energy among members. In other words, to update the share fraction of the whole group based on the new power plant auction, the MTEsm must wait the date the new power plant is ready to operate. Moreover, this function also supports the group crypto-currency market (SEB) since every time a new power plant is approved, some tokens are proportionally created. However, their distribution happens once the whole new power plant implementation process ends. Therefore, independently of the number of beneficiaries, the last step of the *power-up* operation is a transaction that distributes tokens all at once, making the returning values always account for 100% of the new power implemented.

**TRADE** A member can exchange quotas with anyone else by a price agreed (*Figure 19*) upon themselves (an off-chain communication process). Important to

remember that this value is represented by means of the group's token created for this purpose, the SEB. However, the principle of donation is preserved depending on how null values are transacted. If the objective is to donate energy, an exchange of quotas is made with a zero token price. On the other hand, a donation of SEBs has null values of quotas. To keep the group transparency about the quota shares, the MTEsm displays a notification every time the function is triggered.

Now, to continue throughout the reading of the flowcharts, notice what has been described earlier about (i) the meaning of the return arguments at each full operation process; (ii) the notification messages that are displayed; (iii) the persistent storage dynamics (read, write and delete); and (iv) the turning point brought up with off-chain boxes, along with the purpose and behaviour of each operation listed above. The start point is Figure 6, where is clear the conditions to access the three sets of operations and the notification of an invalid request.

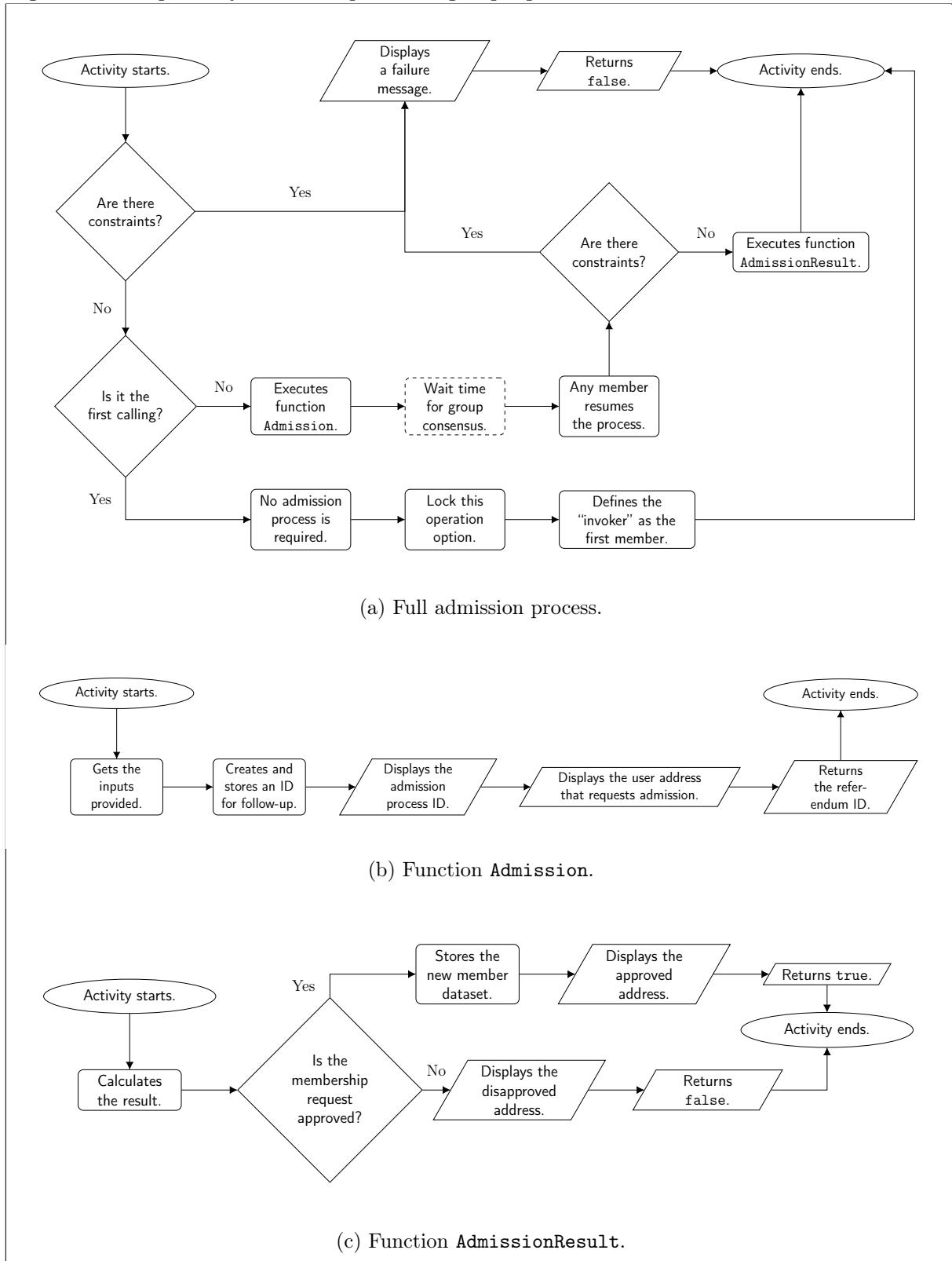
Figure 6 - The flowchart of the MTEsm main interface.



Legend: From here all the operations described before start its performance, that are detailed from Figure 7 to Figure 19.

Source: The author, 2019.

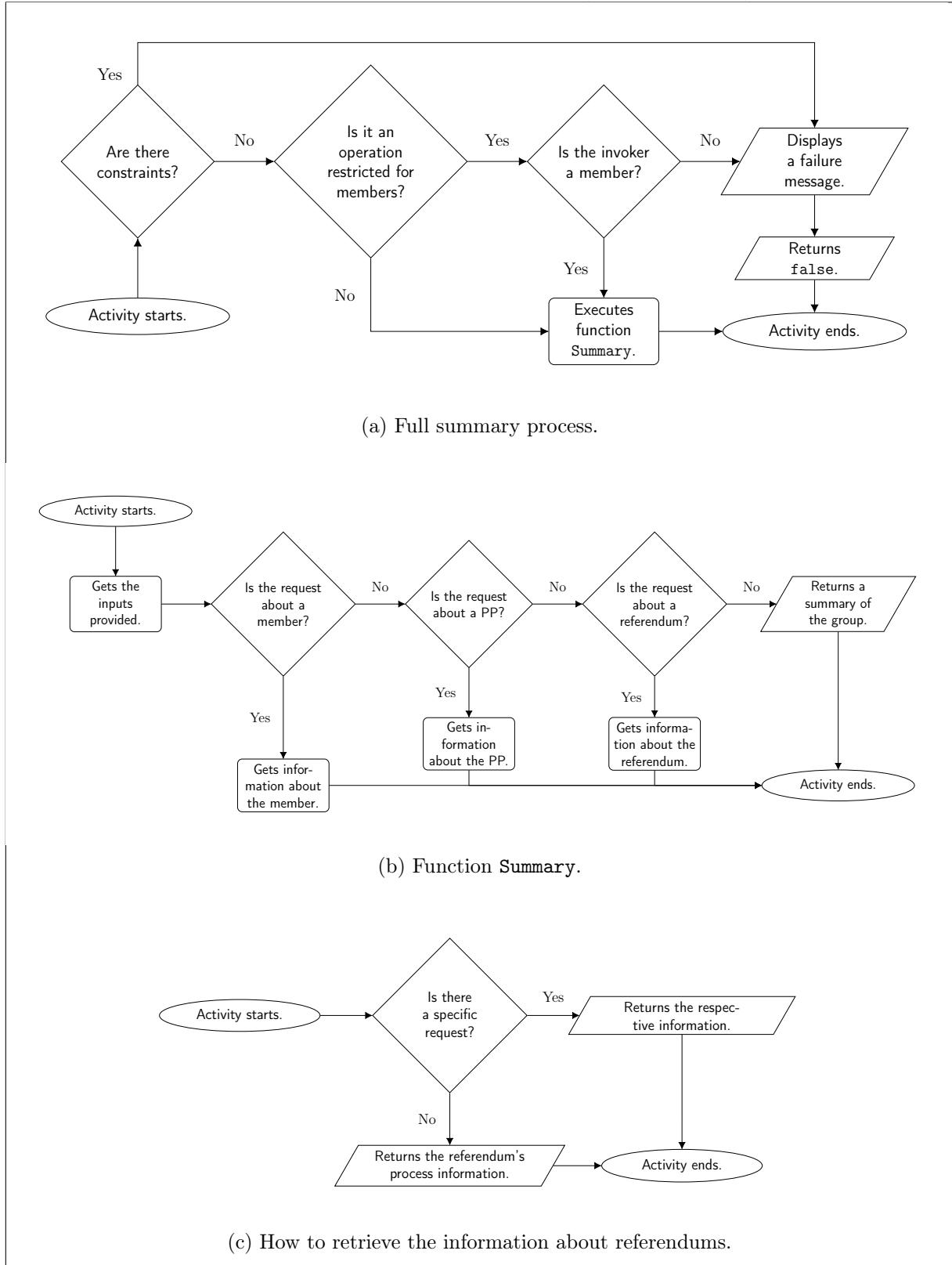
Figure 7 - The pathway to be accepted in a group's private environment.



Legend: Functions (b) and (c) complement the whole process presented on (a).

Source: The author, 2019.

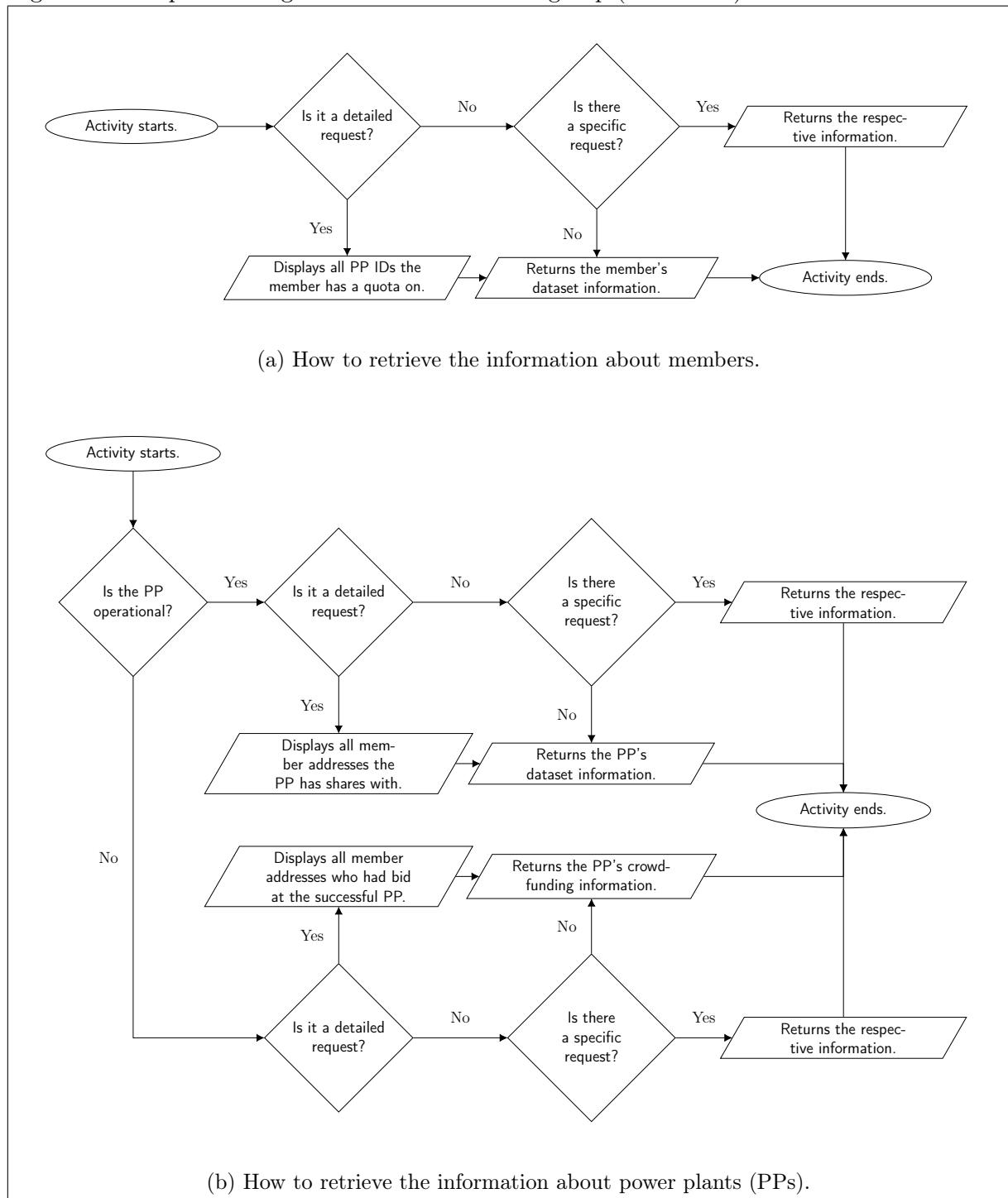
Figure 8 - The process to get information from the group (to be continued).



Legend: Function (b) complements the whole process presented on (a), while (c) is one of the respective steps of (b), that continues in Figure 9.

Source: The author, 2019.

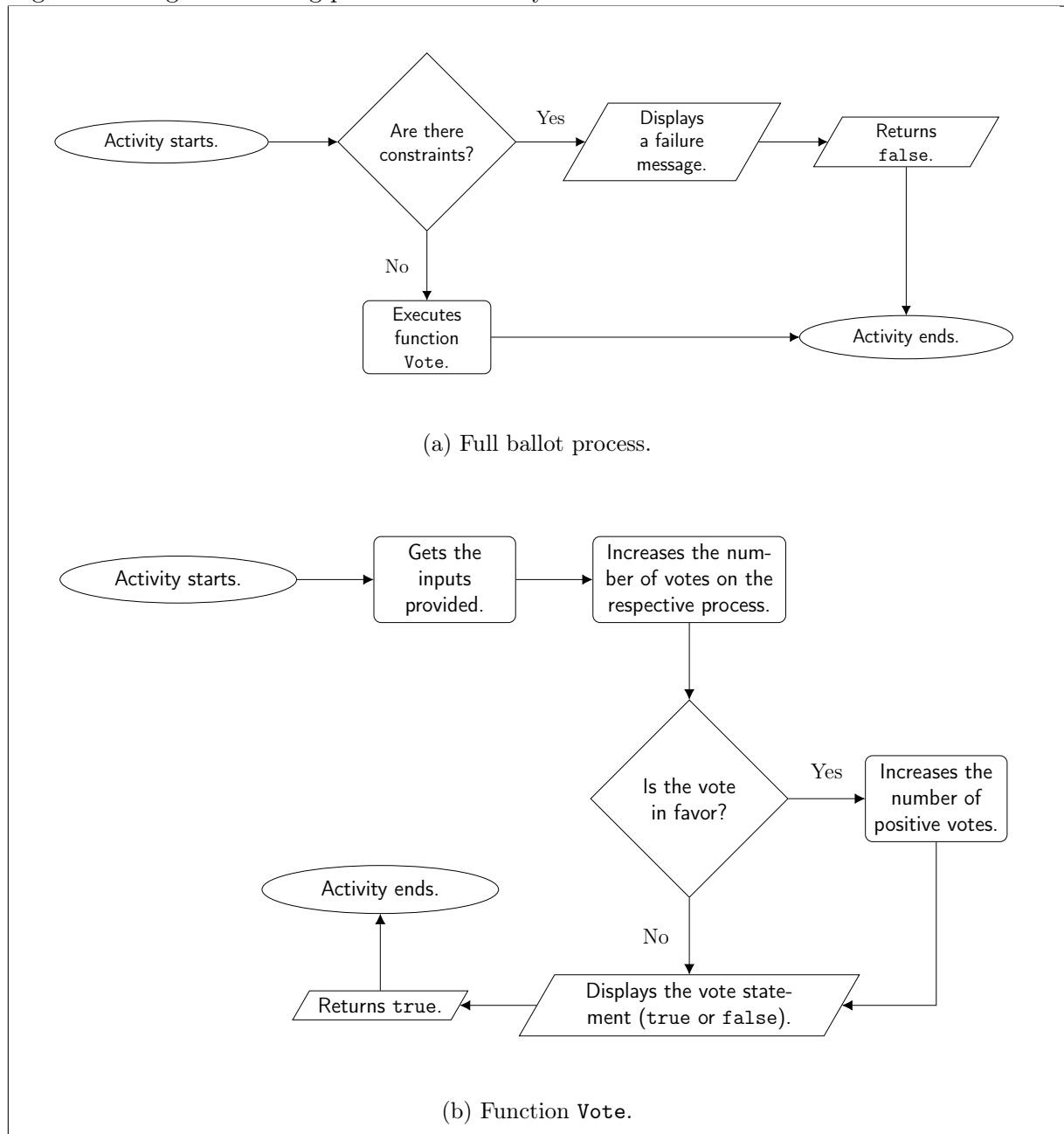
Figure 9 - The process to get information from the group (conclusion).



Legend: Flowcharts (a) and (b) complement the respective steps presented in Figure 8b.

Source: The author, 2019.

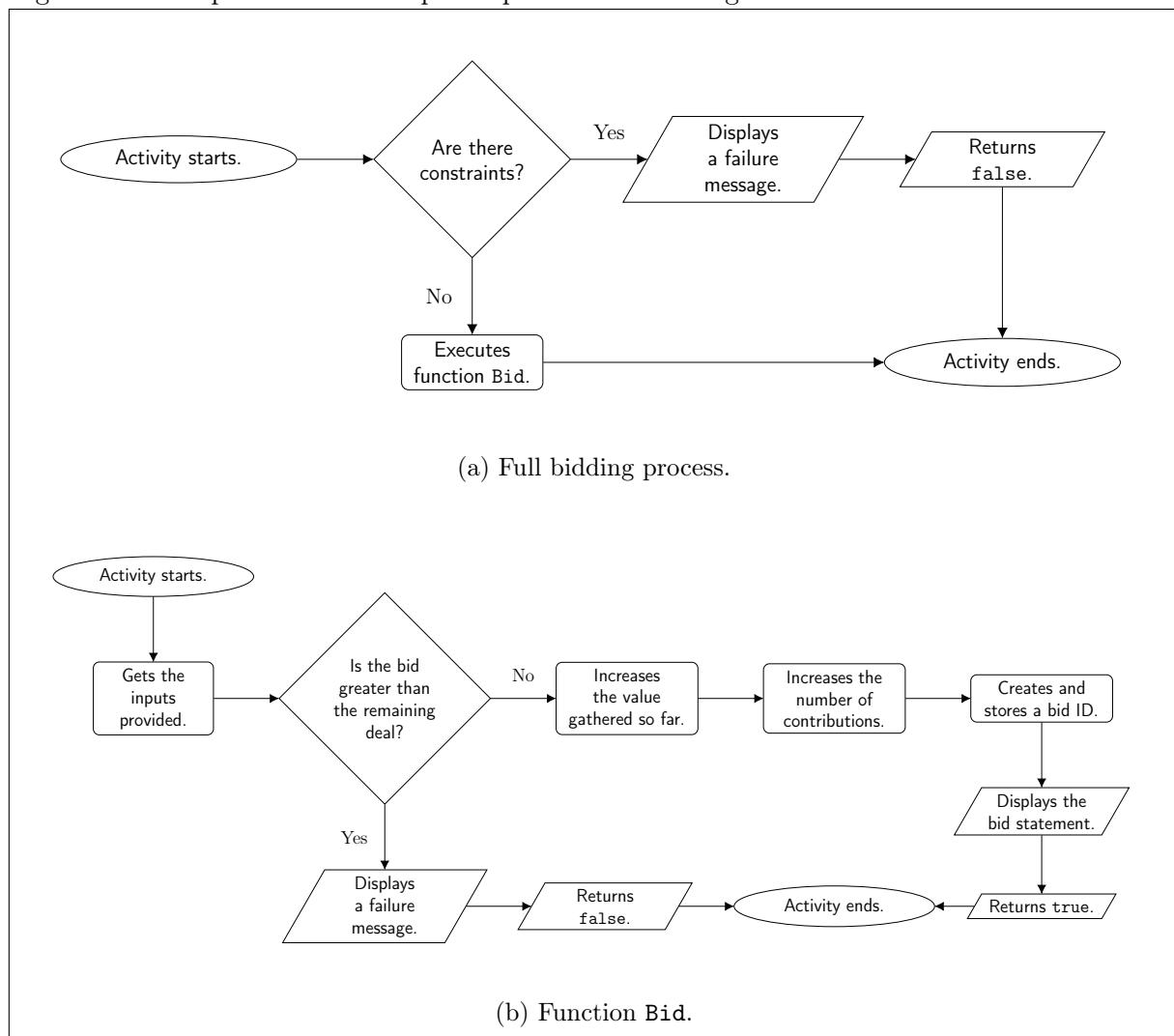
Figure 10 - A general voting procedure called by a member.



Legend: Function (b) complements the whole process presented on (a). Note that the return argument of (b) means the success of the ballot, and not the vote answer.

Source: The author, 2019.

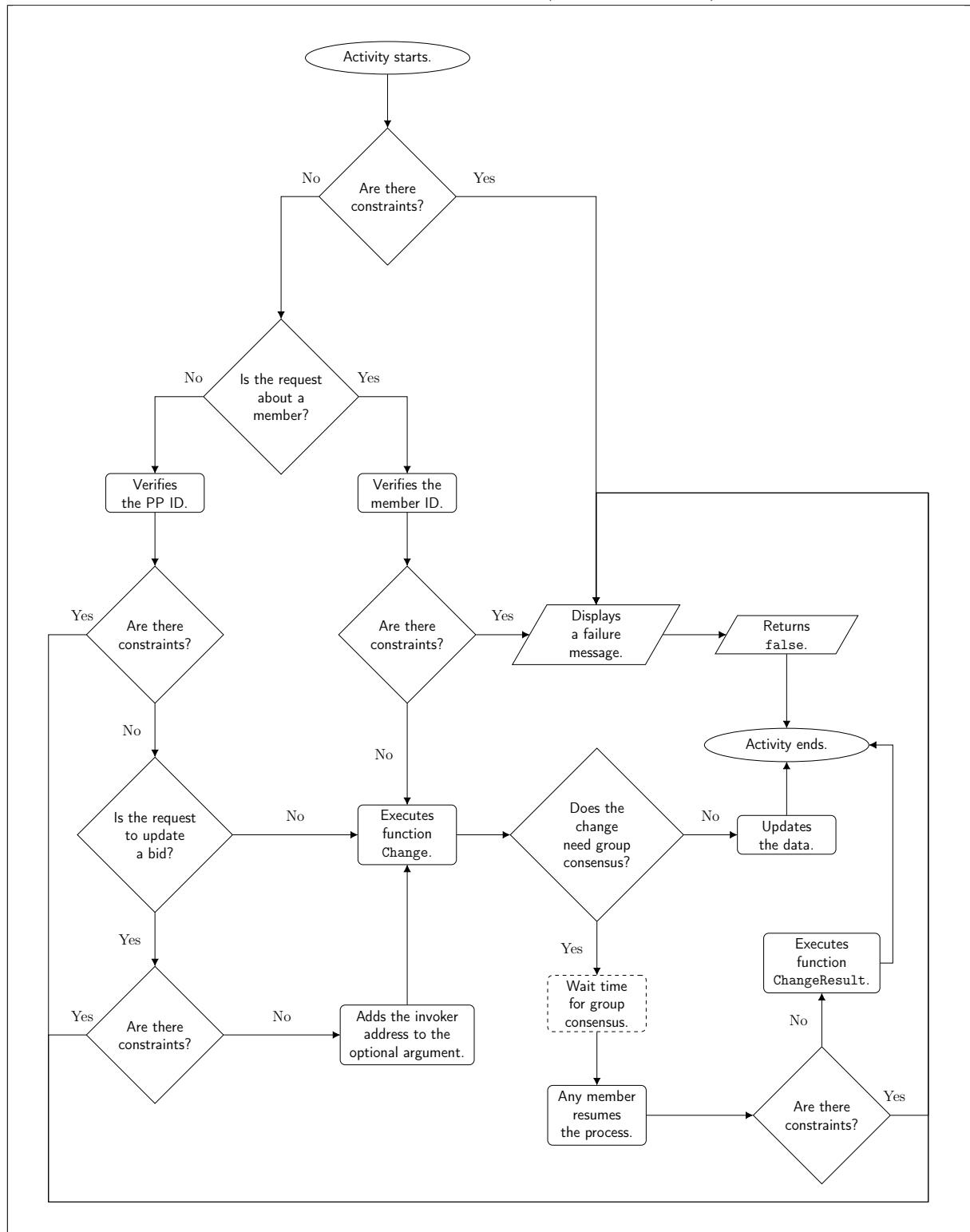
Figure 11 - The process of a new power plant crowdfunding.



Legend: Function (b) complements the whole process presented on (a).

Source: The author, 2019.

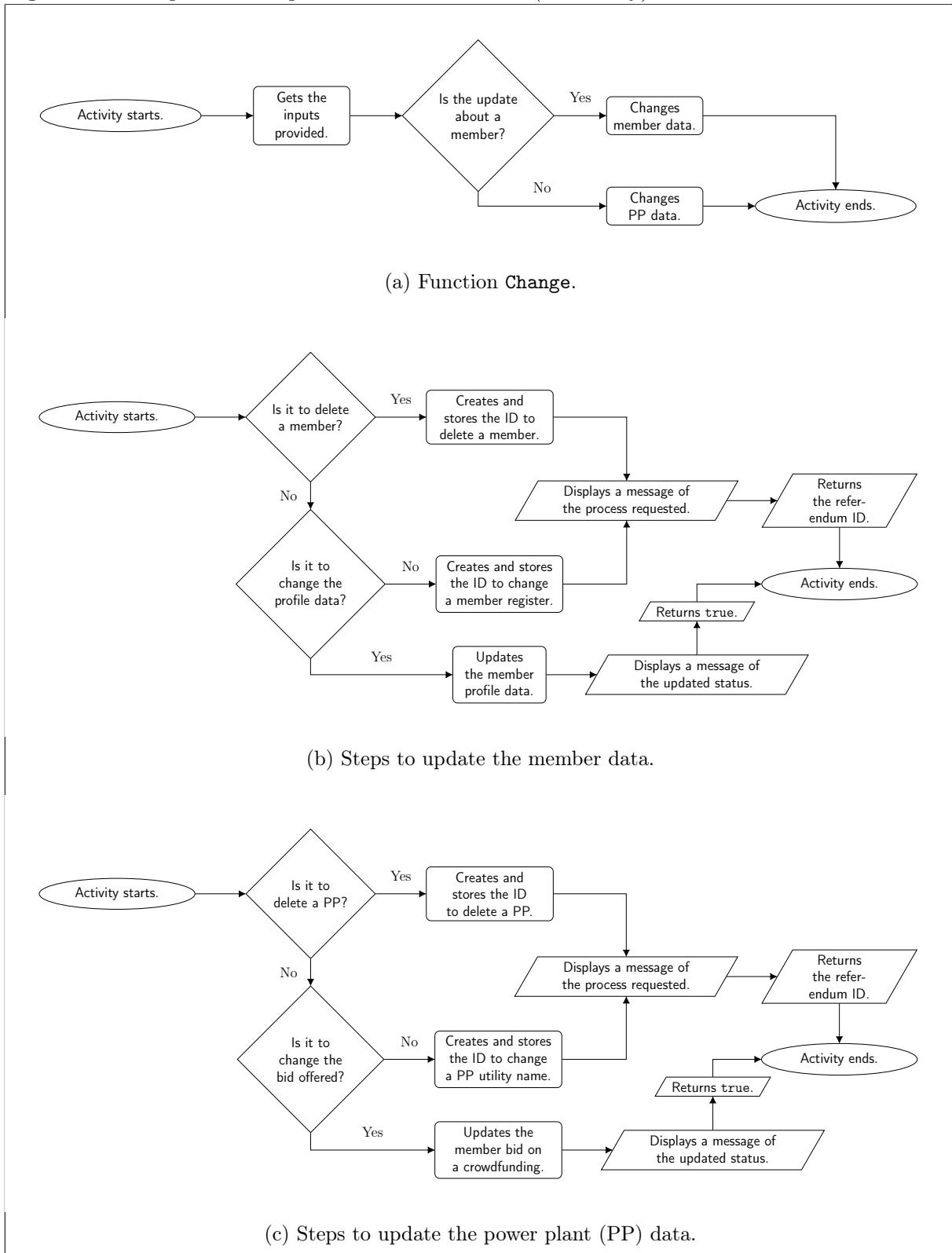
Figure 12 - The process to update some information (to be continued).



Legend: The full updating process of several information. Further details in Figure 13 until Figure 15.

Source: The author, 2019.

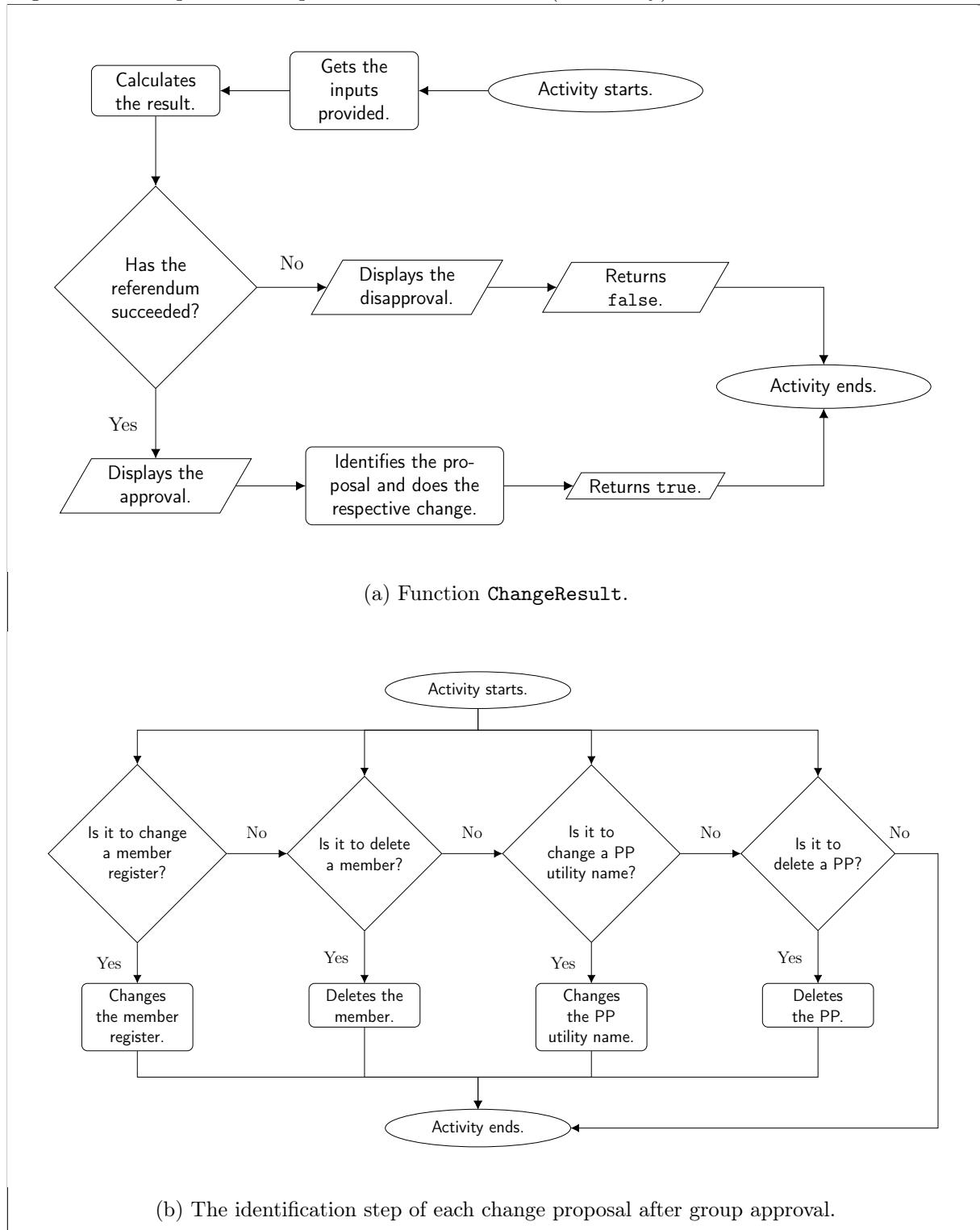
Figure 13 - The process to update some information (continuity).



Legend: Function (a) complements the process presented in Figure 12, while (b) and (c) complement the final steps of (a). Although both have different outcomes, they share a common pattern such as one option to update data without group consensus.

Source: The author, 2019.

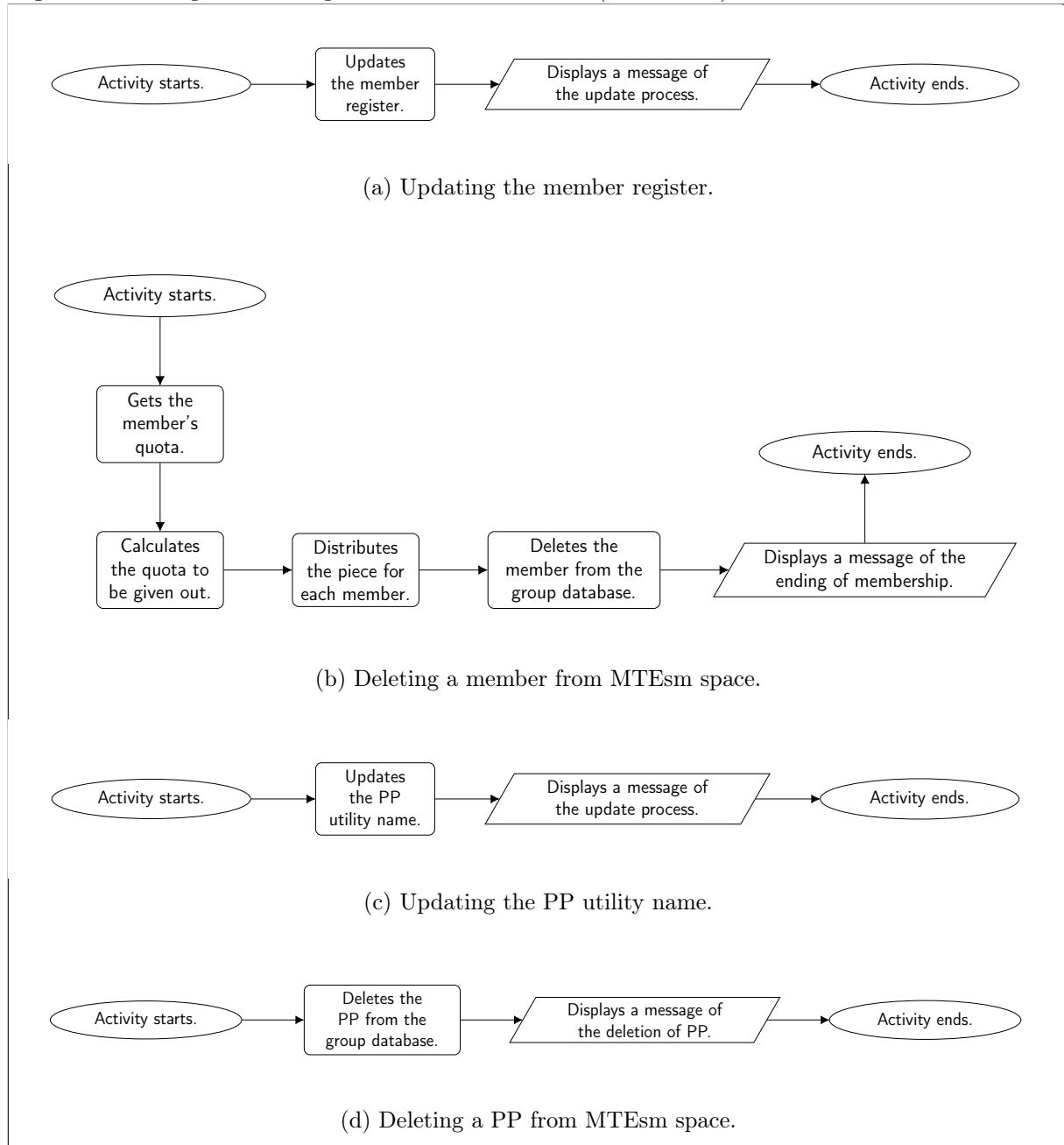
Figure 14 - The process to update some information (continuity).



Legend: The step (b) intermediates the referendum success in the function (a) and the final steps reproduced in Figure 15.

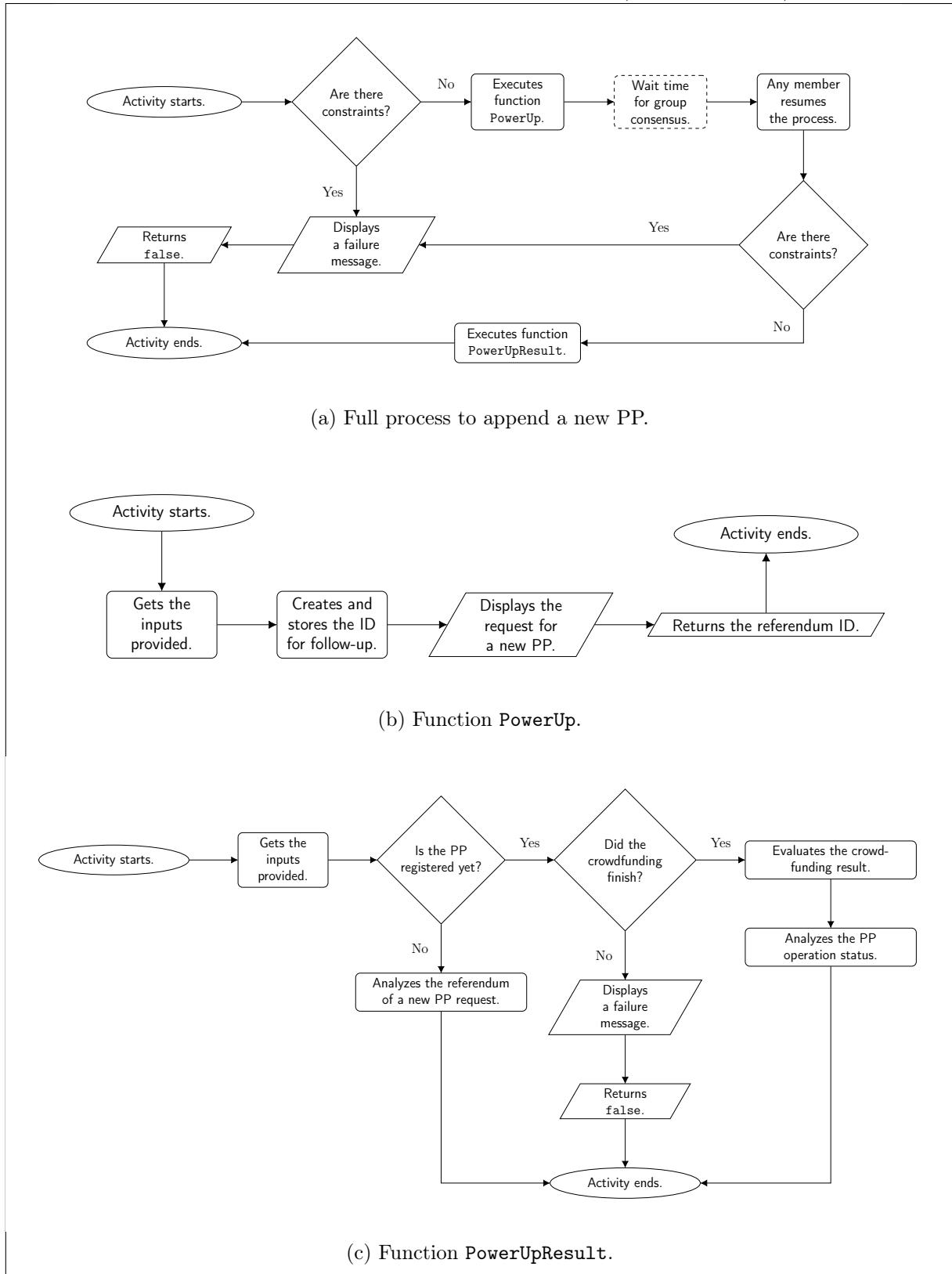
Source: The author, 2019.

Figure 15 - The process to update some information (conclusion).



Legend: The end of the update process accordingly with each proposal identified in Figure 14b.  
Source: The author, 2019.

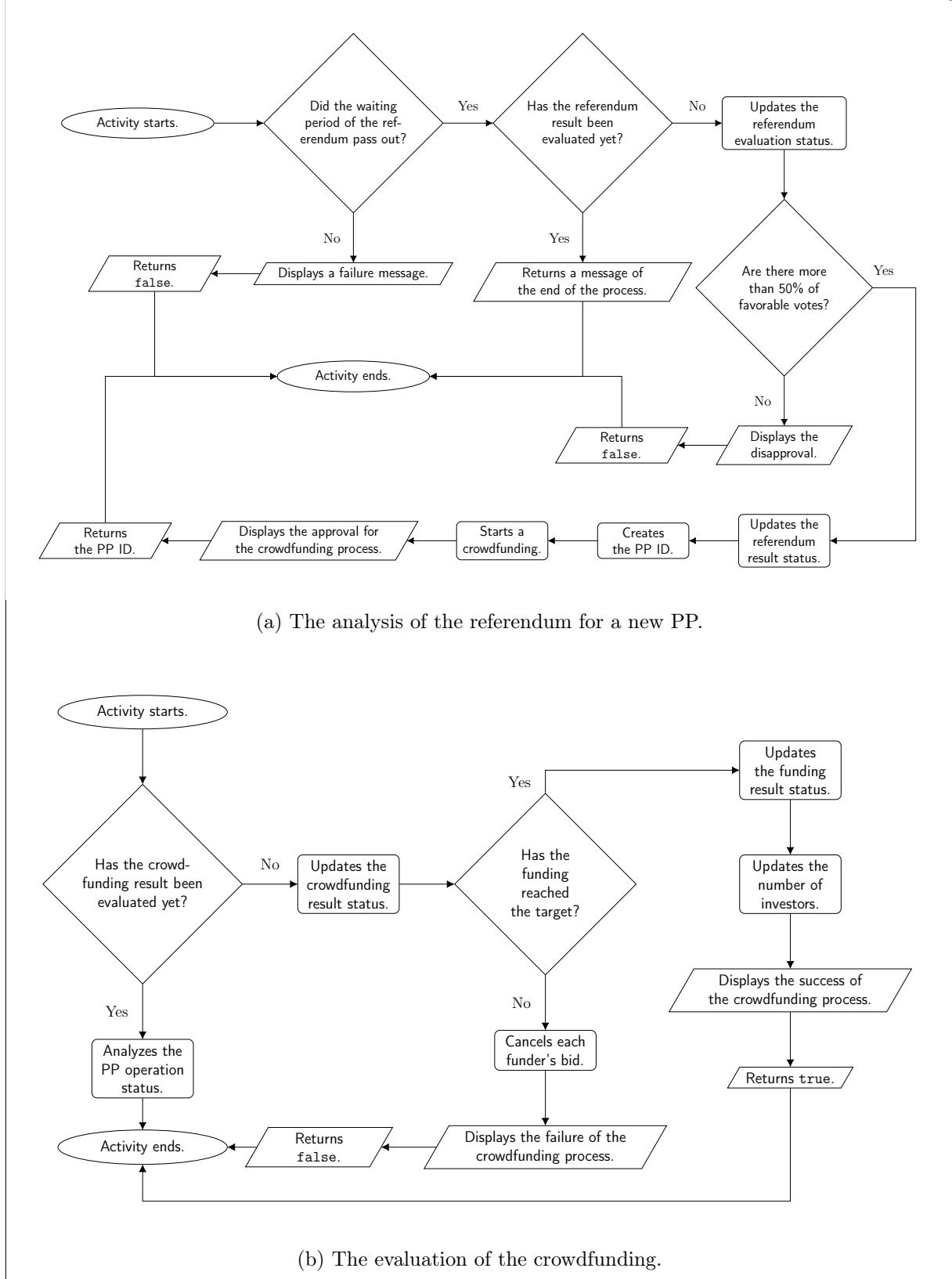
Figure 16 - The process to increase the group power capacity (to be continued).



Legend: Functions (b) and (c) complement the whole process presented on (a).

Source: The author, 2019.

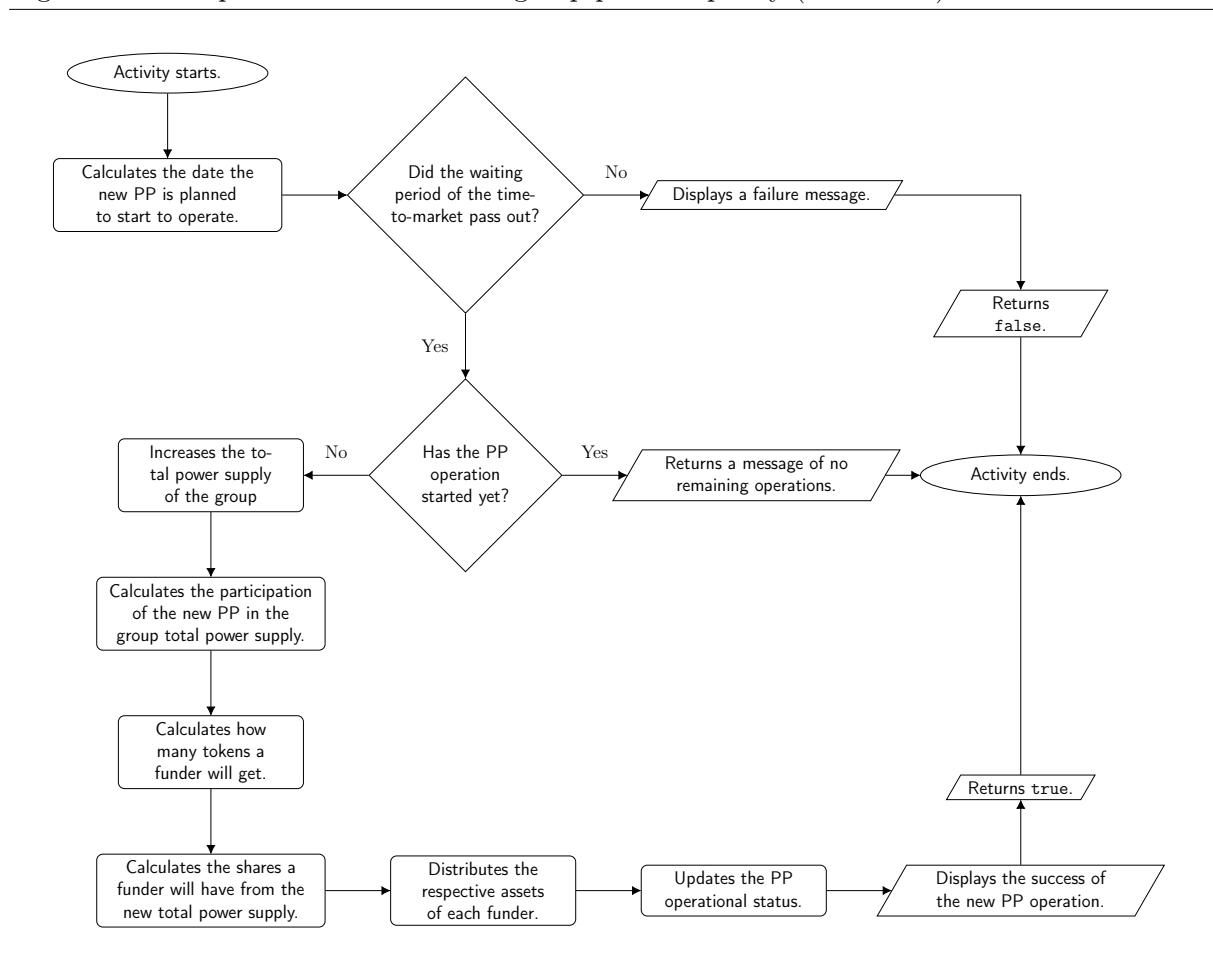
Figure 17 - The process to increase the group power capacity (continuity).



Legend: The steps (a) and (b) complement the initial operations in Figure 16c.

Source: The author, 2019.

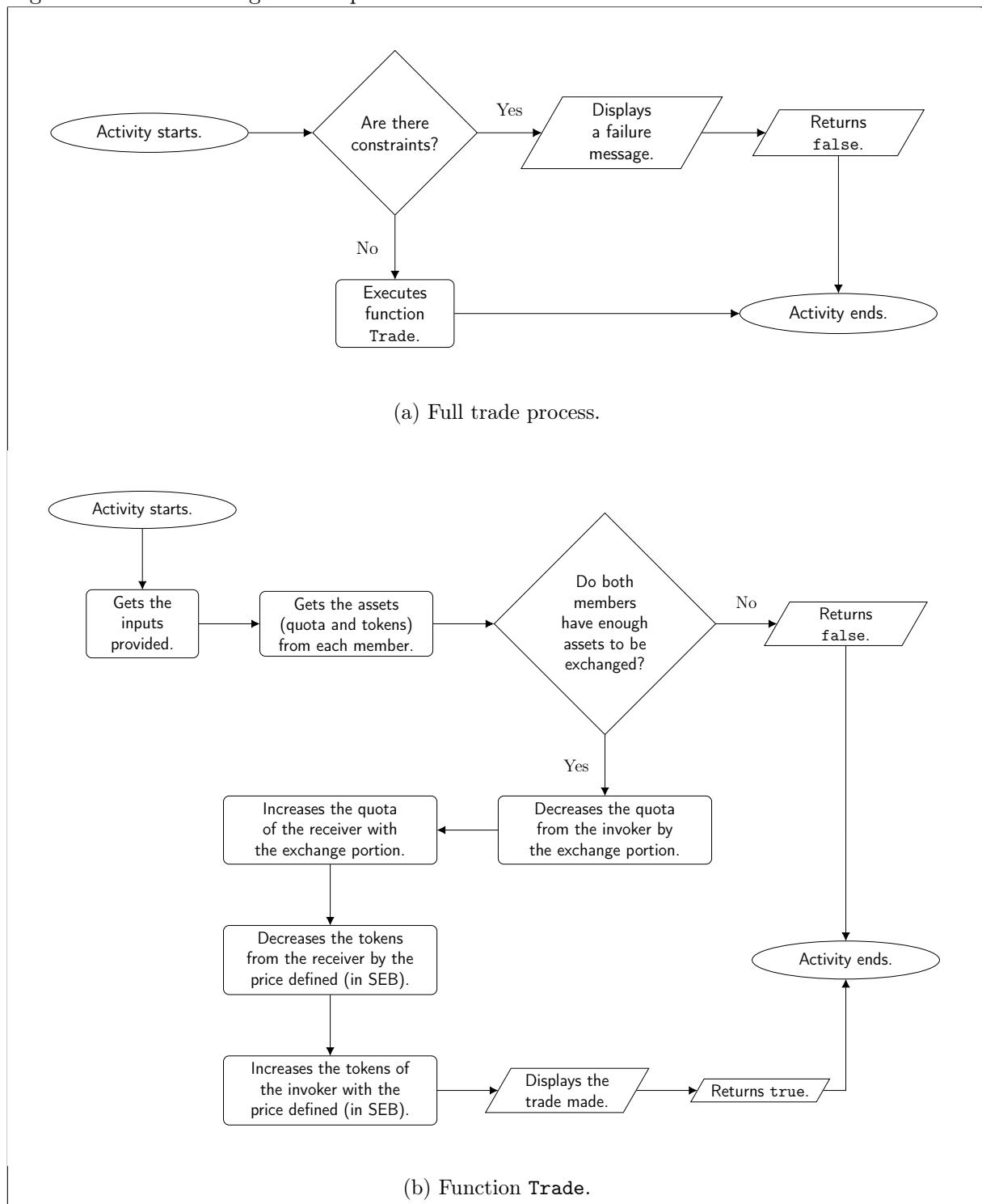
Figure 18 - The process to increase the group power capacity (conclusion).



Legend: The analysis of the PP operation status complements the last step in Figure 16c.

Source: The author, 2019.

Figure 19 - The trade agreement process between members.



Legend: Function (b) complements the process (a).

Source: The author, 2019.

### 3.4 An example of use

The application of the MTEsm is in accordance with the initial example presented in Chapter 1. Therefore, the simplest analysis of the DApp usability considers the set up of three members of the same shareable consumption group belonging to the same power utility coverage area. Remembering that member  $M2$  is a power plant, the member  $M3$  is a prosumer, and  $M1$  is a consumer. To keep the example concise, the register of  $M2$  is left for other opportunity, and the initial steps of the simulation are presented in Appendix F. So, only the trade between members  $M1$  and  $M3$  is in the paragraphs bellow.

As indicated previously in Section 3.2.7, the NeoCompiler test environment (EcoLab) was used to perform the TE test. Also, the smart contract version refers to the one available in Appendix E, even though some adjustments had to be made as indicated in Workarounds (Appendix F).

Therefore, by way of illustration, Figure 20a shows how the member  $M3$  exchange 7 % of her/his quota by 4 SEBs with member  $M1$ . The EcoLab presents a good interface to support the writing of the JSON invoke parameters, that is hidden in the related image, however it is explained at full in Appendix F. Following the descriptions of the *main* and *trade* functions, it is straightforward to pass the required information to get a successful trade operation since each JSON key represents the corresponding variable type required by the MTEsm.

While Figure 20a shows the details to fulfill the input parameters, Figure 20b presents the JSON output with both the blockchain cost of operation and the return arguments of the operation in hexadecimal format. Without financial incentives to fasten the computing (null values of system and network fees) the final cost for the trade was 7.445 GAS paid by  $M3$ . And the returning values of the trade operation follow its function flowchart (Figure 19) with the return argument **True** indicating the success of the operation and the display of a resume of the operation performed as a system notification.

Figure 20 -  $M_3$  sells quotas to  $M_1$ .

hex format of the  $M_3$ 's ID  
(APLJBPhRg2XLhtpxEHd6aRNL7YSLGH2ZL)

Invoke Parameters (JSON)

```
[{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"trade"}, {"type":16,"value": [{"type":5,"value":"23ba2703c53263e8d6e522dc32203339dc8eee9"}, {"type":2,"value":7}, {"type":2,"value":4}]]
```

7%      4 SEBs

hex format of the  $M_1$ 's ID  
(AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y)

(a) Invoke.

```
[{"jsonrpc": "2.0", "id": 5, "result": { "txid": "0x9e869e9ca4ea4a530127759403fb75f1f4cd7480bde766098430542c2b423911", "executions": [ { "trigger": "Application", "contract": "0xf9806d3c45e1adc753871acfda65c17bbad47f2", "vmstate": "HALT", "gas_consumed": "7.445", "stack": [ { "type": "Integer", "value": "1" } ], "notifications": [ { "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0", "state": { "type": "Array", "value": [ { "type": "ByteArray", "value": "7472616e73616374696f6e" }, { "type": "ByteArray", "value": "52eaab8b2aab608902c651912db34de36e7a2b0f" }, { "type": "ByteArray", "value": "23ba2703c53263e8d6e522dc32203339dc8eee9" }, { "type": "Integer", "value": "7" }, { "type": "Integer", "value": "4" } ] } ] } ] } }
```

transaction ID  
(appended on  
the ledger)

computation cost [GAS]

indication of the success  
of the operation (true)

smart contract ID

hex format of “transaction”  
(the system notification)

7%

4 SEBs

(b) Output (JSON RPC).

Legend: The final step of the MTEsm trade operation. The full process is described in Appendix F.

Source: The author, 2019.

### 3.5 Limitations and constraints in the development

In the beginning of this chapter, there were indicated several aspects of a DApp that will not be considered for the MTEsm development. This had allowed to focus on the essential requirements of the smart contract. However, other challenges were encountered during the development course, and now, these ones are addressed to share the learnings and improve future reviews of the smart contract.

Therefore, the immediate attention goes to the different IDs the MTEsm uses, as presented early. The example catches every ID described in Table 5 (Section 3.3) with some of them showcased in Table 6. Since a lot of IDs are created by the MTEsm and required to continue with other operations, these aspects may confuse users, turn the usability more complex, and make the management more challenge to retain notes about all of them. Although, the MTEsm takes advantage of the way blockchain identify its assets to identify several internal operations, the referred smart contract does not have the same mapping structure the blockchains have. Thus, this correlation of data and IDs should be reconsidered for future developments.

Another constraint of the developed DApp is the process to store the address of each member. Actually, it uses an index number for each new member that integrates the group private space. The number increases accordingly with each approved request but it is not a unique identifier since the index value is updated when a member exits the group. Although this implementation works fine by now, it may impact in the performance of the smart contract. The best solution should be a mapping between members and the smart contract, similarly to what happens with the mapping between users and the cryptocurrency NEO. Thus, members could be counted as an asset of the MTEsm and easily tracked in the ledger from the smart contract ID.

A limitation also exist to directly get the address of the *caller* (invoker) of an operation. Currently, the smart contract verifies the *caller* identification in relation to the value of a given variable, as it is stated in the first argument of the *main* function (Code E.6, Appendix E). It may sound strange to write the caller identification twice, i.e., when choosing the invoker and the first variable of the function. Nonetheless, it can be overcome in the front-end development when getting the address of the *caller*. Thus, the same value can be directly passed to the function argument, leaving this variable hidden on the front view but compliant with the smart contract requirement. Due to the basic development of the MTEsm, only the “redundant” caller identification is presented. Its importance can be inspected in several operations through the API function *Runtime.CheckWitness()* to verify user identity, and through the method *GetMemb()* to verify membership status.

A last identified constraint refers to the need of a common database for the utility names. Since it is the variable that links the condition to trade and to get profits of the

energy credits, any misspelling do harm the application purpose. The logic constraints rely on the fact that when the power plant utility name changes in the power plant register, it should update on each member register as well. However, the present smart contract version only allows the member her-/himself to change this information. So, the simple step to update a string has turned into an unreliable operation. On future versions of the MTEsm, it must be fixed carefully.

Table 6 - List of some identifications used along the example.

nº	description
1	<b>ID</b> AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y <b>format</b> Base58 <b>belongs to the</b> EcoLab <b>created at</b> EcoLab configs <b>note</b> Represents the principal fictitious user of the Neo blockchain test environment. It is the default address with NEO, and consequently, able to rapidly gain GAS to perform transactions on the blockchain. It represents the member <i>M1</i> .
2	<b>ID</b> APLJBPhRg2XLhtpxEHd6aRNL7YSLGH2ZL <b>format</b> Base58 <b>belongs to the</b> EcoLab <b>created at</b> EcoLab configs <b>note</b> Other fictitious user of the EcoLab. Used to represent the member <i>M3</i> . Normally, it does not have NEO but can receive GAS from <i>M1</i> to be able to perform the transactions.
3	<b>ID</b> AdgrVTJmPXpfMaq87tzqF7YgKdToBiVQqp <b>format</b> Base58 <b>belongs to the</b> EcoLab <b>created at</b> compilation <b>note</b> It is the smart contract address, that is required for all the interactions with it through transactions.
4	<b>ID</b> dd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d <b>format</b> Hexadecimal (ScriptHash) <b>belongs to the</b> transaction <b>created at</b> the deploy of the smart contract <b>note</b> It is the registering of the MTEsm in the blockchain. It is also searchable in the ledger.
5	<b>ID</b> 4571b1aeb91496ab15111885a9ecbc3967b8cf47385c96795115a5209c1db60a <b>format</b> Hexadecimal (ScriptHash) <b>belongs to the</b> EcoLab ledger <b>created at</b> the deploy of the smart contract <b>note</b> Identifies the block on the chain that contains the transaction described in nº 4.
6	<b>ID</b> RZEYMCoaYLAkPqCpawPgoRJkR5gA3i <b>format</b> Base58 <b>belongs to the</b> MTEsm <b>created at</b> the transaction of the request of <i>M3</i> to join the group <b>note</b> Figure 33b (Appendix F) shows this ID in hexadecimal format (525a45594d436f61594c416b50714370617750676f524a6b523567413369).

Source: The author, 2019.

### 3.6 The group market size

The total amount of tokens allowed is proportional to the mini-grid power capacity limit discussed at Chapter 1 that is equal to 5 MW. By a rate of 1 per 1000, i.e., 1 SEB is created for each 1000 Watts (1 kW) added on the group power capacity, this represents a total value of 5000 SEBs that can be distributed among members. However, small steps should be done until there, and the methodology to do so is presented below.

When identified the need for a new DG installation, whatever if it is the first one or not, the cost of the investment ( $I$ ) and the number of participants must be known in order to properly conduct a crowdfunding process. In Equation 2 (Chapter 1) the member quota ( $q_m$ ) was defined to only one power plant. Now this concept is extended to engage  $n$  investment rounds conducted through the blockchain platform. Therefore, a general representation of a member quota after a successful fund of a  $DG^{(n)}$  is:

$$q_m^{(n)} = \frac{C_m^{(n)}}{I^{(n)}} \quad [\%] \quad (4)$$

The  $C_m^{(n)}$  stands for the contribution made by a member for a *new* power plant, which has a power capacity  $P^{(n)}$  directly proportional to the tokens created by the ratio discussed before. The quota value is also used to determine how much SEBs a member receives from this process round:

$$T_m^{(n)} = q_m^{(n)} \cdot \frac{P^{(n)}}{1000} \quad [\text{SEB}] \quad (5)$$

From both Equations 4 and 5 is easy to find the relationship between fiat and crypto-currency. The member's quota sets the link between these two realms. In one side, the member contribution to build a given power plant  $n$  and her/his respective amount of tokens by a factor of 1000. At the other side, the power capacity of the new power plant and its required investment:

$$\frac{P^{(n)}}{I^{(n)}} = q_m^{(n)} = \frac{1000 \cdot T_m^{(n)}}{C_m^{(n)}} \quad (6)$$

Moreover, the total value of the group's power capacity ( $P^{(T)}$ ) is the sum of each power plant capacity. Similarly happens for the tokens ( $T^{(T)}$ ). Thus, the ratio between power and tokens created keeps the same, which means that members can freely exchange quotas independently of a crowdfunding process or even if they took or not part in it.

$$\begin{aligned} P^{(T)} &= 1000 \cdot T^{(T)} \\ \sum_{x=1}^n P^{(x)} &= 1000 \sum_{k=1}^m T_k \end{aligned} \quad (7)$$

However, a member's quota from a given crowdfunding process ( $q_m^{(x)}$ ) does not represent the share a member has over the group total power capacity because a power plant is just a fraction of it. So, every time a new power plant gets ready to operate, its corresponding shares must be changed to meet the power basis of the group before the update of members' quotas. As a result, it must be known how much a given power plant ( $P^{(x)}$ ) affects the total power capacity ( $P^{(T)}$ ). It can be represented by the factor:

$$f^{(x)} = \frac{P^{(x)}}{P^{(T)}} \quad (8)$$

Thereby, a member's total quota is:

$$q_m^{(T)} = \sum_{x=1}^n (f^{(x)} \cdot q_m^{(x)}) \quad (9)$$

Note that the factor ( $f^{(x)}$ ) does not affect the tokens created, neither distributed. Each member continues to receive her/his tokens due to her/his contribution to fund a given power plant, as described in Equation 5. Then, the total SEBs a member has ( $T_m^{(T)}$ ) is equivalent to how much tokens were acquired during the crowdfunding plus how much was exchanged ( $Ex$ ) with other members:

$$T_m^{(T)} = \sum_{x=1}^n (q_m^{(x)} \cdot T^{(x)}) + \sum_{a=1}^z Ex^{(a)} \quad (10)$$

Where  $T^{(x)}$  is the number of tokens created when a power plant  $x$  is launched. Therefore, independently of the agreements made to exchange SEBs, the methodology will always keep the group total assets account for 100%. And how much a member can profit on this market depends solely on her/his strategy to finance new DGs and trade her/his shares upon the trade-off about her/his own power consumption needs.

### 3.7 The costs of the smart contract's operations

The objective to identify the costs of the operations is to determine the viability of the MTEsm since technical approach seems not to be a problem. However, get the whole DApp cost is a very tough task that is impractical and unnecessary to understand what should and shouldn't be considered when evaluating a blockchain application. So, a basic analysis can be done for the given example presented in Section 3.4. Also, it is worth noting that the following paragraphs perform a rough discussion of the subject, aiming to showcase the values in terms of the order of magnitude and not in precise numbers.

Therefore, although the registering steps of the power plant are missing, the example covers the essential costs to allow the TE. These values replicate the price of each

operation accordingly with the Neo specifications (CELIA, 2019b; CELIA; QIAN, 2019), which is easily identified throughout the whole example course (Appendix F) and is gathered in Table 7.

Table 7 - Financial analysis of the MTEsm.

operation	cost [US\$]	cost [NEO]	system fee [GAS]	network fee [GAS]	payer
registering of 2 users in the blockchain	(2×) 10*	(2×) 1	—	—	<i>M1 &amp; M3</i>
deploy of MTEsm	—	—	990	0.18192	<i>M1</i>
registering of <i>M1</i>	—	—	—	8.212	<i>M1</i>
registering of <i>M3</i> (request)	—	—	—	3.533	<i>M3</i>
registering of <i>M3</i> ( <i>M1</i> 's vote)	—	—	—	4.033	<i>M1</i>
registering of <i>M3</i> (result)	—	—	—	7.549	<i>M1</i>
update of <i>M1</i> 's quota (request)	—	—	—	4.621	<i>M1</i>
update of <i>M1</i> 's quota ( <i>M1</i> 's vote)	—	—	—	4.033	<i>M1</i>
update of <i>M1</i> 's quota ( <i>M3</i> 's vote)	—	—	—	4.033	<i>M3</i>
update of <i>M1</i> 's quota (result)	—	—	—	6.518	<i>M3</i>
update of <i>M3</i> 's quota (full process)	—	—	—	19.205**	1/2
update of <i>M1</i> 's tokens (request)	—	—	—	4.554	<i>M3</i>
update of <i>M1</i> 's tokens ( <i>M1</i> 's vote)	—	—	—	4.033	<i>M1</i>
update of <i>M1</i> 's tokens ( <i>M3</i> 's vote)	—	—	—	4.033	<i>M3</i>
update of <i>M1</i> 's tokens (result)	—	—	—	6.522	<i>M3</i>
trade between <i>M1</i> and <i>M3</i>	—	—	—	7.445	<i>M3</i>
SUBTOTAL	20	2	990	88.5059	
TOTAL	20	2	—	1078.5059	

Legend: \* The price to buy 1 NEO on July 12, 2020. Source: coinmarketcap.com/currencies/neo.

\*\* Considering the right operation and not the pitfall described in STEP 5.2 (Appendix F).

Source: The author, 2019.

At a first glance, Table 7 indicates three different and important instances of the MTEsm financial life cycle. The first is the register of the users in the blockchain. It is susceptible to the world financial market and represents the initial barrier one may consider to use or not the blockchain platform. The second is the cost to register the smart contract in the ledger. It is the only operation configured as system fee and where the big challenge of the smart contract development resides in because the number of lines of code and the kind of operations have a significant impact on the final price. Moreover, as stated in Section 2.1, both errors and exceptions may harm the DApp operation. So, decide upon what and where deal with this code structure also influences in the smart contract development price. The last instance is when users become members of the MTEsm, and all the operations are computed as network fees. At this moment, there is another big challenge since the capacity to perform an operation depends on how much GAS one has, that consequently relies on the amount of NEO has in each member wallet.

Faced with this, and aware that Neo's average time to append 1 block in the ledger is 15 seconds, as indicated in Table 4 (Section 3.2.7), and in the worst scenario (NEO, 2018) only 1 GAS is created by each block, then the Neo blockchain produces 4 GAS/min, consequently 240 GAS/h. It is also known that the distribution of GAS is proportional to how much NEO one has. Therefore, the minimum amount allowed, which is 1 NEO, is equivalent to 0.000001 % of the total amount of NEO available in the network (100 million), and then, 1 NEO guarantees 0.0000024 GAS/h.

As indicated in Table 7, the MTEsm costs 990 GAS to be appended in the ledger. Hypothetically speaking, to reach this value with only 1 NEO is required infinite years of waiting (412,500,000 hours  $\approx$  47,089 years). This turns the cost to be a user of the Neo network not so cheap since the minimum payment available does not allow a feasible outcome. On the other hand, if one has all the 100 million NEOs (which is equivalent to 1 billion US dollars<sup>29</sup>), the waiting period will be slightly more than 4 hours.

Although the analysis could stop here because the DApp price is already impracticable, let suppose one more hypothetical step to evaluate the dynamism of the energy intra-market. Considering both members with half of the maximum amount of NEOs, i.e., each one with 50 million NEOs, then each member will get 495 GAS in 4.125 hours. Thus, they both reach 990 GAS in 8.25 hours, and right after, any member can deploy the DApp and start to perform the TE operations.

For simplicity, while one member pays the system fee, the other transfers half of her/his GAS to the other through the Neo wallet, so both can equally make transactions with the MTEsm. Therefore, each member has 495 GAS at this moment. It allows them to immediately perform the full process after the deploy operation alone since it requires less than 89 GAS. However, as presented in Table 7, the member *M1* covered 8 out of 17 operations, i.e., 47 % of the total cost of the network fee except for the trade operation cost. In the end, one could say that members *M1* and *M3* still have 453.17 GAS and 436.83 GAS, respectively.

From this point, the other operations that may arise – with no further registering of power plants, either of new members – will turn the trade between members the most valuable operation, which is extensively affordable due to its low cost (7.445 GAS). Moreover, if the members' GAS does not increase over time, each one may respectively request up to 60 and 58 transactions. Therefore, if the members would like to exchange their quotas once a month, they could do so during the next 5 years.

However, likewise the cost of the MTEsm deploy, the group capability to pay for each transaction is not manageable in time, either in resources. Since the analysis had considered 50 million NEOs for each member, which corresponds to 500 million US

---

<sup>29</sup> Accordingly with the conversion rate indicated in Table 7.

dollars<sup>29</sup>, the costs to pay for the transaction in a reasonable time is still high. Even though there is no need for a fast transaction rate due to the group decision requirements happen in a monthly-based time frame, as stated before in Section 3.2.7, with little NEOs one has a very long waiting time to be able to perform an operation.

Finally, the group market size structure and dynamism (as presented in Section 3.6) may benefit from the operations showcased by the present analysis, even with the block-chain platform price being prohibitive for a large DApp. Certainly, further economic studies of other available platforms must be considered to properly allow a tempting energy marketplace for medium/low voltage power consumers.

## CONCLUSION

The main goal of the current study was to create a smart contract to enable the Transactive Energy (TE) for Brazilian micro/mini-grids. Several steps were taken to present a quasi-POC since the lack of discussion with stakeholders does not permit a complete study. However, the importance of this publication follows general efforts to integrate blockchain applications into the electricity sector. Moreover, it also aligns with the prevailing subject of sustainable alternatives to reach the worldwide growing demand for energy.

In a fast pace, the following paragraphs review the study course with the main concepts presented until shows the outcomes that can be used for future works. The study road map goes from the understanding of the needs of energy management alternatives inside micro/mini-grids, to considerations of different use cases of the blockchain technology in the energy sector. Then, after the comprehension of the blockchain technology and its different platforms available, a DApp prototype was designed with a unique smart contract to test the solution proposed. This academic-only outcome enabled an evaluation of the smart contract development, a partial feasibility of blockchain integration with current business models and several new questions to discuss further.

### **The study summary**

Therefore, while the TE term may sound new, its purpose have been discussed for some years, and the emerge of the blockchain technology has turned its objective much more achievable. However, as indicated in Introduction, the literature presents some challenges to integrate ICTs into power grids towards a more intelligent system, such as the smart grids purposes. Besides the discussions to properly manage the grid under this circumstances, the blockchain may bring a new thinking of the current energy market and business models operation.

Since the aforementioned concepts are not an isolated case, Chapter 1 identified several studies and projects in development in every part of the world. Their applications range throughout the whole electricity chain with innovative solutions of payment, trading, asset management, energy control, and renewable energy tracing. Mapping these projects helped to establish where the present study lays down and to define its contribution for the TE studies.

Likewise other blockchain applications for the management of energy in the distribution grid do, this study also considered a specific regulation framework and a use case scenario. For DGs under Brazilian rules, an example was introduced to showcase

how a group of shareable consumption may manage their own micro/mini-grid composed of residential and commercial power units. The simplistic example setup with different consumer profiles, and only three members enabled to demonstrate how the blockchain technology may overcome suspicious over a central authority and lack of transparency in the management of the group's assets.

Beyond the energy sector, several blockchains and other Distributed Ledger Technologies (DLTs) exist to fulfill different market requirements. For the following years, a significant impact are expected on how businesses will manage their assets. Indeed, several discussions have arose about what should be considered an asset. Front of such kinds of uncertainties and expectations, a reasonable understanding of the blockchain technology was made in Chapter 2 to guide the presented development throughout a right direction at least.

Thereby, the blockchain Neo was chosen as the development platform in consideration to the trade-offs between other blockchains available. It was noted that the performance requirements were with minor importance to the final decision since all the platforms under analysis use almost the same kind of consensus algorithm. Likewise, the provenance, immutability and finality of the operations in the ledger have a very similar behaviour and the throughput has little interference in the outcomes for this case in particular. On the other hand, the condition to get better access to pay for the transactions, to evaluate a test application, and to store values in a private space were decisive to choose the blockchain Neo.

Aware of the opportunity to work on and the fundamental platform to develop a possible solution, Chapter 3 has identified some software requirements to support the smart contract development. It was clear that this process must involve a lot of stakeholders and discussions far beyond this study could reach but the efforts made were enough for the time being.

Following the ISO/IEC/IEEE International Standard of Systems and software engineering – Life cycle processes – Requirements engineering (2011), there were identified a handful of information to be considered or not part of the DApp developed. In summary, several decisions to conduct economic operations and to avoid pitfalls when interacting with the smart contract were taken into consideration. But a lot of the usability requirements and user experiences subjects, that need integration with both off-chain operations and other platforms, were kept off the development.

So, in a broader view, the concepts of the system were detached into three layers where one is related to the power network, other to the business network and another to the information network. The latter was the target of the study with the blockchain platform being the link between both remaining layers. It was also identified that the peers of this network is indeed the members of the group of shareable consumption, i.e., the members are the only ones with capacity to read at and to write in the ledger. On the

other hand, both the power utility and government representatives can only read general information about the group as ordinary users of the blockchain.

Next, the methodology to manage the group's energy assets and exchanges were defined. The practical principle of the sharing of energy proportionally to each member contribution to finance a micro/mini-grid power plant has not been modified and was the basis for the methodology. Nonetheless, the novelty of this kind of crowdfunding was to perform the registering of the auction processes in the blockchain. But the payment and charge of bids are off-chain operations, which were not covered in the development as stated before.

Moreover, following the ICO concept, the related methodology is also used to create the group's unique crypto-currency Sharing Electricity in Brazil (SEB). This token represents the group's energy currency and is used to define how much of energy a member has rights over the group's power capacity. Therefore, an intra-market was established wherein members may acquire tokens either financing a power plant or trading with other members.

Finally, all of these processes gave rise to the Microgrid Transactive Energy smart contract (MTEsm). Although it is merely a small part of a complete DApp, it is the core of a blockchain application with remarkable details to care about. For simplicity, the MTEsm is a single smart contract developed in C# with a number of operations to manage the intra-market, to consult the quotas, to participate democratically to the group's decisions, and so on. All of this with transparency and different levels of access to the group's reports. However, a lot of difficulties and challenges were identified and several aspects of the DApp solution could be evaluated as well. These outcomes are discussed in the following sections.

## Presentation of findings

The quasi-POC was possible due to the identification of the core concepts of the application requirements as indicated in Figure 4 (Section 3.1). Even missing a lot of important information, there were encountered learnings in great numbers. These are classified in the three sets of knowledge below, which follows the development scopes discussed before. A last section complements with the remarks of the problems also found in the development of the MTEsm.

## External Environment

Most of the contributions to this requirement level were summarized in the beginning of this chapter since it is tightly correlated with the descriptive analysis. The following key findings abstract this discussion of market trends, laws & regulations, technology base, standards & specifications, public culture, and physical/natural environment:

- (1) the shareable consumption group is the unique DG framework regulated by the Brazilian law where consumers and prosumers may be get together and determine its own guidelines to share the expenses of an alternative power source besides the energy offered by power utilities;
- (2) an easy-to-use test environment to evaluate the smart contracts is vital to any DApp development. The condition to collaborate with others and save the code increase a lot the development satisfaction. The NeoCompiler Eco was a differential for the study; and
- (3) get the support from standards, best practice guides and awareness of other projects had significantly contributed to prevent further hurdles in the development course.

## Business Operation

In this level, some processes and constraints were defined and identified:

- (4) the MTEsm use case has demonstrated the importance to consider a detailed trade-off analysis about what data and computation should be placed on- and off-chain to define the best blockchain fit, as indicated in Table 2 (Section 2.3);
- (5) the understanding of the crypto-currencies creation rate helped to examine the expenses between the blockchains which is translated into how many read/write operations can be done in a period;
- (6) likewise, the feature of a side database to store specific values of a smart contract has a debatable and undetermined cost. In the current scenario it has no fees for the Neo blockchain but can do so in the future. Moreover, the kind of information and the size of the file may interfere in the final cost to use this feature. And then, new analysis of redundancy and reliability should be made to set the requirements of storage and access for a given file;
- (7) the clarification that a smart contract development should be language-agnostic and follow software development best-practices showed most more relevant than to

choose the blockchain platform due to a particular programming language. In the end, identify the requirements and respective costs to keep the application working well is more important than the programming skills;

- (8) the decision for a democratic structure based in equal votes instead of a members' hierarchy was important to showcase one of the blockchain features for transparent and auditable processes;
- (9) however, the multi-signature feature could also be used for the same purpose but further studies of the blockchain development environment must be considered;
- (10) the discussion of the smart contract's economy highlighted the challenge one may have to balance how much crypto-currencies are required to support the expenses with read/write operations;
- (11) the exchange of tokens is enabled or not according to each power utility the members belong to. This follows the Brazilian DG rules, as well as the limits of capacity of the power plants;
- (12) besides the expected trade operation between members, the options to donate either energy or SEB were also implemented. The three operations are performed by the same function but with different invoke methods to deal with each option available;
- (13) some variables that are dependent on off-chain operations were developed to be adjusted along with real-time planning of the group daily life. For instance, the time required to update the group power capacity, and consequently update the energy shares of the members, depends on the date the new power plant starts to operate. The MTEsm is deployed with constant values for *waiting time* but some of them should be changed. Figure 22 shows where this variables interfere in the processes (Appendix C);
- (14) the crowdfunding option to pay for the DG creates a unique intra-market for the group. On the one hand, the crypto-currencies created reflect the group power capacity. On the other hand, each member only commit with its capability to pay for a given energy fraction. Therefore, the advent of new consumption patterns, due to family increase/decrease or comfort investment, may demand an update either in the group power capacity or solely in the member shares. This unpredictability dictates when it is time to create new SEBs or to exchange it;
- (15) also, it was considered that every contribution for a crowdfunding must remain registered even if the process fails or is cancelled. This keeps the integrity and transparency for whatever happens in the decision of the group power capacity;

- (16) even though some MTEsm operations depends on off-chain platforms, not all operations may be synchronized with on-chain procedures. For instance, the registering of bids in a crowdfunding process has a time frame well defined by the smart contract. However, the payment for this commitment may happen in a different time frame that is dependent on the off-chain platform chosen for this purpose.

## System Operation

At this last level, it is presented the findings related to the development of the DApp, and mainly the contributions with the MTEsm:

- (17) the definition of the member as the only one with capacity to either read and write in the group private space (Figure 3, Chapter 3) has enabled the identification of a vulnerability in the system that cannot be overcame by only one stakeholder, neither by the blockchain technology solely. Since the registering in the ledger of the power generation and consumption depends on the values measured by the energy meter, much more discussion must be done to address this issue;
- (18) the tentative implementation of restricted access to consult the MTEsm private information (Figure 6, Section 3.3.1) was unsuccessful due to the API option to retrieve the information from any transaction performed;
- (19) although the members' agreements and the power measurements registered in the ledger can be used for legitimacy, the requirement for off-chain integrations still depends on trust factors of intermediary applications;
- (20) other aspect of the code structure is the caution with the cost of each function. It is preferable to have a longer algorithm with a lot of statements to evaluate the need to perform or not a given function, instead to pay for a failure operation;
- (21) although the smart contract can be written in C#, the NeoVM does not support the full library of the language. This makes the development process more challenging since a lot of grounded concepts should be relearned to comply with the particularities of the platform;
- (22) the difficulties to get the address of the invoker straight in for an operation variable was surpassed with the replication of the value in the variable box. However, when developing a complete DApp, the front-end must overcome this duplicated alternative;

- (23) the IDs of the MTEsm follows a similar pattern creation of the one used by blockchains. This has raised knowledge about internal specifications of the technology related to encoding/decoding, cryptography and computing limitations;
- (24) the implementation of exceptions gave better usability and financial alternatives to perform operations. This follows the principles of blockchains where all programmable computation is charged. The MTEsm tried to take care of this condition to get the most of the benefits of the distributed systems, as indicated in Appendix D;
- (25) also, the MTEsm follows the Neo standard of time format to deal with all constants and variables that represents a period;
- (26) moreover, several notifications were properly set to enable easy integration with other APIs when interacting with the JSON output. This follows the best practices of blockchains applications too;
- (27) the smart contract has different configuration to delete each kind of the group's data. To delete information of a member, the MTEsm redistribute the quotas for the remaining members. However, the elimination of a power plant information does not affect the members' data since their quotas is a percentage rate of the group power capacity, and not of the single value of a power plant capacity. This difference is visually indicated in Figure 15b and Figure 15d (Section 3.3.1);
- (28) the MTEsm was developed to conduct the user through a good experience to reach the desired outcome or at least to understand what got wrong. This avoid unwanted waste of *GAS* and better comprehension of the constraints;
- (29) as succinctly showed in Table 7 (Section 3.7), the deploy of the MTEsm might be prohibitive, specially if payed by only one user. But the transactions fee are paid by whoever request the operations and these costs are directly dependent on the user ownership of *GAS*. The methodology to get enough credit to pay for the operations remains a huge challenge;
- (30) as indicated by the financial analysis of the code, it is better to validate the variables format with off-chain methods than to pay for an invalid operation. Dealing with mistakes is expensive in the blockchain.

## Development pitfalls

This section aims to briefly highlight the problems faced with the MTEsm development. For sure, a bigger example than the presented here will show more points for

improvements, and several tests still need to be made to catch other ones. So far, it is possible to add to the number of contributions:

- (31) the definition of some names in the notification messages and in the variables should be reviewed to better correspond to its purpose, like as choose “trade” instead of “transaction” to indicate the trade transaction (Figure 20b, Section 3.4);
- (32) should also review what is considered sensitive data since the APIs of public block-chains will always keep the output information accessible to everyone;
- (33) fix the use of different instances to register the utility names of both members and power plants. This raise a lot of problems to update a diverse set of information in the MTEsm;
- (34) the function that counts the number of votes is wrong. Fortunately, it was caught during the example but did not affect the desired outcome (Figure 44, Appendix F). The function should be fixed to consider the group consensus of more than a half of the total number of participants to take any effect, and not bigger or equal than it, as currently done;
- (35) the operation to validate a user as a member requires special attention too since the increase of members turns the voting process more complex with different permissions to participate or not for a given referendum. Similar operations that depends on the voting process may experience similar unwanted behaviour. This may result on operations lock in time and a new method may overcome theses situations;
- (36) the MTEsm IDs are extremely valuable to the smart contract functioning but take notes of all of them is challenging and may compromise the DApp usability;
- (37) the process to store the address of each member follows a dynamic index that may impact in the performance of the smart contract. A better alternative should be a mapping between members and the smart contract, indicating that a user has became an asset of the MTEsm which can be tracked by an specific operation; and
- (38) the limitations to perform some operations at full when deployed as part of the MTEsm gave rise to a new release of the smart contract version. These slight modifications were tenderly nicknamed WAR, as indicated in the end of Appendix F.

## **Future directions**

This study has brought up many questions in need of further investigation. For instance, why is so hard to implement ICT into power systems? Does the off-chain

integrations, such as get data from power metering, is really a point-of-failure due to current digital measurement available? In other to evaluate a testbed, what consumer classes should be considered first? Low power consumers to analyze micro investments and fewer exchanges or big consumers to get an overview about fast ROIs and heated marketplace? Or maybe a mix of both? Additionally, for a complete POC analysis, how a blockchain platform can impact the application usability due to a possible change on its pricing model? Can new regulation procedures on the energy sector wreck the benefits brought by blockchain integration?

Besides the questions, some specific subjects may guide a path to follow. For example, nowadays the classical mechanical devices used for power metering are being substituted for a digital device with telemetry measurement and control. Although this option represents an improvement on energy management and finer power measures for the distribution utility, it can still present failures on site and on remote computers. What concerns for digital issues, such as the difference between values gathered on site and those displayed on remote places, it can be overcome with a directly integration of blockchain with power meters. A difficult task that can strengthen the reliability for whoever depends on the metering values to conduct a new business. Indeed, this technical subject is relevant for academia too due to the properties of accuracy, traceability, security and privacy for the data under consideration, which are essential to ensure reliable energy markets in a future time-based tariff (VANGULICK; CORNÉLUSSE; ERNST, 2018).

Moreover, great attention to the price of the blockchain operations is required because this component is vital to forecast how many and what kind of operations can be made in a given period. As presented in the Comparison between the blockchains (Section 3.2.7), there are different ways to charge for the interactions with the blockchain. Some of the details are at the smart contract level to avoid needless operations, and others at business decision level to join or not a blockchain infrastructure platform. Understand the financial realm that underpins those distributed computing networks will leverage the development of DApps to a new paradigm.

Another pillar of the POC is the electricity sector regulation. The group structure, the payment rules, the government fiscal incentives and many more legal considerations must be addressed for a complete proposition since a slight change on any of this subjects will impact the presented market model. Notably, the blockchain alternative to establish an agreement for a payment transaction is an avid discussion, even though crypto-currencies has already been regulated for trading operations in several countries.

One last emphasis is about the user experience to deal with the variables of a DApp. The Base58 format turns the values much more friendly but it is still hard to get notes about them. Find a pleasant way to take care of private and public keys, processes' IDs and crypto-currencies expenses goes beyond what the trendy crypto wallets do.

## Publishing outcomes

Initial discussions about the present theme was shown at *XLIX Simpósio Brasileiro de Pesquisa Operacional* under the special session of Operational Research and Smart Cities on August 2017. The article *Tendências para Sistemas Microgrids em Cidades Inteligentes: uma visão sobre a Blockchain* (GABRICH; COELHO; COELHO, 2017) represents the first publication experience towards the full development of the master course.

Right after, a single invite to contribute to the publishing of the mentors at the book *Smart and Digital Cities* in the chapter *When CI and Decentralized Systems Effectively Meet Smart Cities and Grids* (COELHO et al., 2019) with a discussion about micro/minigrid implementations on future cities and how blockchain could integrate this sector and assist other important city-like subjects.

Lastly, the smart contract developed here is also available at the online repository GitHub for a worldwide open discussion and collaboration with who else is interested in the subject. It is accessible at [yurigabrich.github.io/microgrid-dapp](https://yurigabrich.github.io/microgrid-dapp).

## Final remarks

*De facto*, the blockchain opens doors for novel business alternatives in the electricity sector but the costs to do so is still expensive. Moreover, it is still required to take much more attention to security aspects when developing DApps, from the smart contracts design to the user experience interfaces. However, the conditions to share costs and benefits are immeasurable and can diminish suspicion in every business decision step.

For what concerns the Transactive Energy (TE) objective, the number of remaining questions are countless. They range from business and regulatory subjects, to technical specifications of both power and computer aspects. In addition, the discussions of the power grid management options – centralized and decentralized – will last forever but the economy of scale is imperative to determine when it is time to change from one option to another, or at least to better integrate them.

Surely, the proposed model is not exhaustive and must be complemented with and questioned by colleagues. Even with so many questions already presented, there are still a lot of ones concerning the basic requirements to start a hands-on approach. Gather really interested people from academia, government and industry to go further on this subject might reveal more interesting questions to be discussed on and to overcome the challenges presented so far.

Maybe a financial trade-off between profits and available risks may answer most of the aforementioned questions. Maybe the real tough task is evaluate them. Hopefully, the present work will contribute to explain the power sector for an interdisciplinary audience

and to incentive the integration of blockchains in much more areas. This will certainly be a fruitful area for further multidisciplinary and collaborative work.

## REFERENCES

- ACKERMANN, T. et al. Paving the way: A future without inertia is closer than you think. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2017. v. 15, n. 6, p. 61–69. ISSN 1540-7977. Available at <<https://dx.doi.org/10.1109/MPE.2017.2729138>>.
- Agência Nacional de Energia Elétrica. *Geracão Distribuída*. [S.l.]: Superintendência de Regulação dos Serviços de Distribuição (SRD), 2016. Online. Available at: <<http://www.aneel.gov.br/geracao-distribuida>>. Accessed on March 17, 2017.
- \_\_\_\_\_. *Perguntas e respostas sobre a aplicacão da Resolução Normativa nº 482/2012*. Brasília/DF, Brasil, 2016. Available at: <[http://www2.aneel.gov.br/arquivos/PDF/FAQ\\_GD\\_Atualizado.pdf](http://www2.aneel.gov.br/arquivos/PDF/FAQ_GD_Atualizado.pdf)>. Accessed on March 24, 2017.
- \_\_\_\_\_. *A Tarifa de Energia Elétrica*. [S.l.]: Superintendência de Gestão Tarifária (SGT), 2016. Online. Available at: <<http://www.aneel.gov.br/tarifas>>. Accessed on Mar 24, 2017.
- \_\_\_\_\_. *Banco de Informações de Geracão (BIG)*. 2017. Online. Available at: <[http://www.aneel.gov.br/informacoes-tecnicas/-/asset\\_publisher/CegkWaVJWF5E/content/big-banco-de-informacoes-de-geracao/655808](http://www.aneel.gov.br/informacoes-tecnicas/-/asset_publisher/CegkWaVJWF5E/content/big-banco-de-informacoes-de-geracao/655808)>. Accessed on June 13, 2018.
- \_\_\_\_\_. *Bandeiras Tarifárias*. [S.l.]: Superintendência de Gestão Tarifária (SGT), 2017. Online. Available at: <<http://www.aneel.gov.br/bandeiras-tarifarias>>. Accessed on Apr 5, 2017.
- \_\_\_\_\_. *Entendendo a Tarifa*. [S.l.]: Superintendência de Gestão Tarifária (SGT), 2017. Online. Available at: <[http://www.aneel.gov.br/entendendo-a-tarifa/-/asset\\_publisher/uQ5pCGhnyj0y/content/composicao-da-tarifa/654800](http://www.aneel.gov.br/entendendo-a-tarifa/-/asset_publisher/uQ5pCGhnyj0y/content/composicao-da-tarifa/654800)>. Accessed on Mar 24, 2017.
- \_\_\_\_\_. *Modalidades tarifárias*. [S.l.]: Superintendência de Gestão Tarifária (SGT), 2017. Online. Available at: <<http://www.aneel.gov.br/tarifa-branca>>. Accessed on Apr 18, 2017.
- ALEIXO, G. *O potencial transformador da tecnologia blockchain*. Rio de Janeiro, RJ, Brasil: Institute for Technology and Society of Rio de Janeiro (ITS Rio), 2016. Online. Available at: <<https://feed.itsrio.org/o-potencial-transformador-da-tecnologia-blockchain-777a5edf6a29>>. Accessed on March 22, 2017.
- BEHLENDORF, B. Hyperledger and the blockchain opportunity. In: BLOCKCHAIN E A TRANSFORMAÇÃO DIGITAL, 2019, Centro de Convocações, CENPES, Cidade Universitária da UFRJ, Rio de Janeiro, RJ, Brasil. Online: Hyperledger, The Linux Foundation, 2019. Available at <<https://www.hyperledger.org/resources/webinars>>.
- BINANCE RESEARCH. *First Look: Libra*. 2019. Online. Available at: <<https://info.binance.com/en/research/marketresearch/libra.html>>. Accessed on Nov 5, 2019.
- BITCOIN.ORG. 2019. Online. Available at: <<http://bitcoin.org>>.

- BLACKSHEAR, S. et al. *Move: A Language With Programmable Resources*. Online, 2019. Available at: <<https://developers.libra.org/docs/assets/papers/libra-move-a-language-with-programmable-resources.pdf>>. Accessed on November 5, 2019.
- BROWN, R. G. *The Corda Platform: An Introduction* White Paper. Online, 2018. Available at: <<https://www.r3.com/white-papers/>>.
- BUTERIN, V. *On Silos*. [S.l.]: Medium, 2014. Online. Available at: <<https://blog.ethereum.org/2014/12/31/silos>>. Accessed on November, 4 2018.
- CASTRO, N. de et al. *Impactos Sistêmicos da Micro e Minigeração Distribuída*. Rio de Janeiro, RJ, Brasil, 2018. (Texto de Discussão do Setor Elétrico (TDSE), 79). Available at: <<http://www.gesel.ie.ufrj.br/index.php/Publications>>. ISBN 978-85-93305-46-7.
- CELIA. *Smart Contract Writing Basics*. [S.l.]: Neo, 2019. Online. Authors' name were deduced from the list of contributors on the platform. Available at: <<https://docs.neo.org/docs/en-us/sc/write/basics.html>>. Accessed on July 13, 2019.
- \_\_\_\_\_. *System Fees*. [S.l.]: Neo, 2019. Online. Authors' name were deduced from the list of contributors on the platform. Available at: <<https://docs.neo.org/docs/en-us/sc/fees.html>>. Accessed on Apr 22, 2019.
- CELIA; QIAN, J. *Charging Model*. [S.l.]: Neo, 2019. Online. Authors' name were deduced from the list of contributors on the platform. Available at: <[https://docs.neo.org/docs/en-us/tooldev/concept/charging\\_model.html](https://docs.neo.org/docs/en-us/tooldev/concept/charging_model.html)>. Accessed on July 13, 2019.
- COELHO, V. N.; COELHO, I. M.; CARDOSO, F. H. *EcoLab*: Neo ecosystem laboratory. 2018. Online. An initiative of the NeoCompiler Eco(system) supported by the NeoResearch. Available at: <<https://neocompiler.io/>>.
- COELHO, V. N. et al. A communitarian microgrid storage planning system inside the scope of a smart city. *Applied Energy*, 2016. ISSN 0306-2619. Available at: <<http://doi.org/10.1016/j.apenergy.2016.12.043>>.
- \_\_\_\_\_. Multi-objective energy storage power dispatching using plug-in vehicles in a smart-microgrid. In: RENEWABLE ENERGY. Online: Elsevier, 2016. v. 89, p. 730–742. Available at <<http://doi.org/10.1016/j.renene.2015.11.084>>.
- \_\_\_\_\_. Multi-agent systems applied for energy systems integration: State-of-the-art applications and trends in microgrids. In: APPLIED ENERGY. Online: Elsevier, 2017. v. 187, p. 820–832. ISSN 0306-2619. Available at <<http://dx.doi.org/10.1016/j.apenergy.2016.10.056>>.
- \_\_\_\_\_. When CI and Decentralized Systems Effectively Meet Smart Cities and Grids. In: \_\_\_\_\_. *Smart and Digital Cities*. Online: Springer, 2019. (Urban Computing), p. 239–249. ISBN 978-3-030-12254-6. Part IV. Available at <[https://doi.org/10.1007/978-3-030-12255-3\\_15](https://doi.org/10.1007/978-3-030-12255-3_15)>.
- Câmara de Comercialização de Energia Elétrica. *Comercialização*. 2017. Online. Available at: <[www.ccee.org.br/portal/faces/pages\\_publico/onde-atuamos/comercializacao](http://www.ccee.org.br/portal/faces/pages_publico/onde-atuamos/comercializacao)>. Accessed on March 17, 2017.

- \_\_\_\_\_. *Como se dividem:* Categoria de agentes da CCEE. 2017. Online. Available at: <[https://www.ccee.org.br/portal/faces/pages\\_publico/quem-participa/como\\_se\\_dividem](https://www.ccee.org.br/portal/faces/pages_publico/quem-participa/como_se_dividem)>. Accessed on March 27, 2017.
- \_\_\_\_\_. *Fontes.* 2017. Online. Available at: <[www.ccee.org.br/portal/faces/pages\\_publico/onde-atuamos/fontes](https://www.ccee.org.br/portal/faces/pages_publico/onde-atuamos/fontes)>. Accessed on March 17, 2017.
- \_\_\_\_\_. *Setor elétrico.* 2017. Online. Available at: <[https://www.ccee.org.br/portal/faces/pages\\_publico/onde-atuamos/setor\\_eletrico](https://www.ccee.org.br/portal/faces/pages_publico/onde-atuamos/setor_eletrico)>. Accessed on March 17, 2017.
- DIGITAL CURRENCY INITIATIVE. *Blockchain Applications to Solar Panel Energy: Landscape Analysis.* [S.l.]: Massachusetts Institute of Technology (MIT), 201-. Online. Available at: <[http://dci.mit.edu/assets/papers/15.998\\_solar.pdf](http://dci.mit.edu/assets/papers/15.998_solar.pdf)>. Accessed on August 2, 2018.
- ELA, E. et al. Electricity markets and renewables: A survey of potential design changes and their consequences. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2017. v. 15, n. 6, p. 70–82. ISSN 1540-7977. Available at <<http://doi.org/10.1109/MPE.2017.2730827>>.
- Empresa de Pesquisa Energética. *Demanda de Energia 2050.* Rio de Janeiro, RJ, Brasil, 2014. (Série Estudos da Demanda de Energia). Nota técnica DEA 13/14. Ministério de Minas e Energia (MME). Available at: <<http://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes>>. Accessed on March 30, 2018.
- \_\_\_\_\_. *Inserção da Geração Fotovoltaica Distribuída no Brasil:* Condicionantes e impactos. Rio de Janeiro, RJ, Brasil, 2014. (Série Recursos Energéticos). Nota técnica DEA 19/14. Ministério de Minas e Energia (MME). Available at: <<https://bit.ly/34y9fjH>>. Accessed on October 2, 2017.
- ENDEAVOR. *Blockchain: conheça a tecnologia por trás da revolução das moedas virtuais:* Proinfa. 2015. Online. Available at: <<https://endeavor.org.br/blockchain/>>. Accessed on March 3, 2017.
- FARHANGI, H. The path of the smart grid. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2009. v. 8, n. 1. ISSN 1558-4216. Available at <[https://doi.org/10.1109/MPE.2009.934876](http://doi.org/10.1109/MPE.2009.934876)>.
- FILIPPO, D. D. R.; SZTAJNBERG, A. *Bem-vindo à Internet.* Rio de Janeiro, RJ, Brasil: Editora Brasport, 1996. Online. Available at: <<http://www.filippo.eti.br/livro>>.
- FORFIA, D.; KNIGHT, M.; MELTON, R. The view from the top of the mountain: Building a community of practice with the gridwise transactive energy framework. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2016. v. 14, n. 3, p. 25–33. Available at <<https://doi.org/10.1109/MPE.2016.2524961>>.
- FREITAS, R. B. de; GOULART, R. P. R.; GABRICH, Y. B. *Estudo para avaliação das condições de operação elétrica do Sistema Interligado Nacional (SIN) frente ao impacto da conexão de geração distribuída na rede de distribuição de eletricidade.* 127 p. Final Project (Bachelor degree) — Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ), Rio de Janeiro, Brasil, jan 2014.

GABRICH, Y. B.; COELHO, I. M.; COELHO, V. N. Tendências para sistemas microgrids em cidades inteligentes: uma visão sobre a blockchain. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL (SBPO), 49th., 2017, Blumenau, SC, Brasil. Online: SOBRAPO, 2017. p. 3934–3945. Available at <<http://www.sbpo2017.iltc.br/sessoes-especiais-sesp1.html>>.

GIANELLONI, F. et al. The distributed electricity generation diffusion impact on the brazilian distribution utilities. In: INTERNATIONAL CONFERENCE ON ENERGY AND ENVIRONMENT: BRINGING TOGETHER ENGINEERING AND ECONOMICS, 3rd., 2017, Porto, Portugal. Online: Faculdade de Economia da Universidade do Porto (FEP), 2017. Available at <<https://goo.gl/5bmqU3>>.

GOGERTY, N.; JOHNSON, P. Network Capital: Value of Currency Protocols Bitcoin & SolarCoin Cases in Context. Columbia Business School, n. 19-2, Oct 2018. Available at: <<http://dx.doi.org/10.2139/ssrn.3281845>>.

GROARKE, D. *Utilities and blockchain pilots*: a global snapshot. [S.l.]: Indigo Advisory Group, 2016. Online. Available at: <<https://www.indigoadvisorygroup.com/blog/2016/10/7/utilities-piloting-blockchain-application-globally>>. Accessed on September 24, 2017.

HAHN, P.; MIGHELÃO, T. R. *The Brazilian Market Of Distributed Solar PV Generation*: Annual report 2017. [S.l.], 2017. Instituto para o Desenvolvimento de Energias Alternativas na América Latina (IDEAL Institute) and Câmara de Comércio e Indústria Brasil-Alemanha (AHK-RJ). Available at: <[https://issuu.com/idealeco\\_logicas/docs/estudoidealmercadofv2017\\_en\\_web](https://issuu.com/idealeco_logicas/docs/estudoidealmercadofv2017_en_web)>. Accessed on October, 2018.

HESSE, E. *Energy Market Transformation*. Online: GBH Forum Network, 2016. Video. Available at: <<https://www.youtube.com/watch?v=cpMwPhA9QzM&feature=youtu.be&t=1h22m11s>>. The Grid Singularity's trading presentation starts at 1h 33m 5s. A description about the whole event is also available at <<https://www.mitforumcambridge.org/event/the-blockchain-enabling-a-distributed-and-connected-energy-future/>>.

HOGAN, W. W. Market design practices: Which ones are best? [in my view]. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2019. v. 17, n. 1, p. 100–104. Available at <<https://doi.org/10.1109/MPE.2018.2871736>>.

HOMMELBERG, M. P. F. et al. A novel architecture for real-time operation of multi-agent based coordination of demand and supply. In: IEEE POWER AND ENERGY SOCIETY GENERAL MEETING, Conversion and Delivery of Electrical Energy in the 21st Century., 2008, Pittsburgh, PA, USA. Online: IEEE, 2008. p. 1–5. ISSN 1932-5517. Available at <<https://doi.org/10.1109/PES.2008.4596531>>.

HUSEBY, D.; PIEKARSKA, M. *Blockchain is the New Black: What about enterprise security?* Online: Hyperledger, The Linux Foundation, 2018. Webinar. Available at: <<https://www.hyperledger.org/resources/webinars>>. Accessed on Dec 7, 2018.

HYPERLEDGER. *Hyperledger Architecture*: Smart contracts. Online, 2018. v. 2. Available at: <<https://www.hyperledger.org/resources/publications>>.

\_\_\_\_\_. *Hyperledger Architecture*: Introduction to hyperledger business blockchain design philosophy and consensus. Online, 2018. v. 1. Available at: <<https://www.hyperledger.org/resources/publications>>.

Instituto Brasileiro de Geografia e Estatística. *Projecão da população do Brasil e das Unidades da Federação*. 2018. Online. Available at: <<https://www.ibge.gov.br/apps/populacao/projecao>>. Accessed on June 13, 2018.

INTEL CORPORATION. *PoET 1.0 Specification*. 2018. Online. (Sawtooth Version 1.1.2). Available at: <<https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html?highlight=proof%20elapsd%20time>>. Accessed on February 4, 2019.

International Energy Agency. *Technology Roadmap: Smart grids*. Paris, France, 2011. 52 p. Available at: <<https://webstore.iea.org/technology-roadmap-smart-grids>>. Accessed on January 30, 2018.

\_\_\_\_\_. *Energy Access Outlook 2017*: from poverty to prosperity. Paris, France, 2017. Available at: <<https://www.iea.org/reports/energy-access-outlook-2017>>.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Systems and software engineering: Life cycle processes, requirements engineering. *ISO/IEC/IEEE International Standard — 29148:2011(E)*, Online, p. 1–94, 2011. Available at: <<https://doi.org/10.1109/IEEESTD.2011.6146379>>.

JIAYI, H.; CHUANWEN, J.; RONG, X. A review on distributed energy resources and microgrid. In: *RENEWABLE AND SUSTAINABLE ENERGY REVIEWS*. Online: Elsevier, 2008. v. 12, n. 9, p. 2472–2483. ISSN 1364-0321. Available at <<http://dx.doi.org/10.1016/j.rser.2007.06.004>>.

JING. *Practical Tips in Developing Neo Smart Contracts*. [S.l.]: Medium, 2018. Online. Authors' name were deduced from the list of contributors on the platform. Available at: <<https://medium.com/@gongxiaojing0825/practical-tips-in-developing-neo-smart-contract-a872a4f910c1>>. Accessed on May 2, 2019.

KANDASWAMY, R. *Blockchain: How Real Is The Market?* Online: Gartner, 2016. Webinar. Available at: <<https://www.gartner.com/webinar/3607520>>. Accessed on April 11, 2017.

LEW, D. et al. The power of small: The effects of distributed energy resources on system reliability. In: *IEEE POWER AND ENERGY MAGAZINE*. Online: IEEE, 2017. v. 15, n. 6, p. 50–60. ISSN 1540-7977. Available at <<https://dx.doi.org/10.1109/MPE.2017.2729104>>.

LIBRA. *White Paper*. 2019. Online. Available at: <<https://libra.org/en-US/white-paper/>>. Accessed on Nov 5, 2019.

LINUX FOUNDATION. *Blockchain for Business*: An Introduction to Hyperledger Technologies. Edx platform: LinuxFoundationX, 2017. Massive Open Online Course (MOOC). Available at: <<https://www.edx.org/course/blockchain-business-introduction-linuxfoundationx-lfs171x>>.

- LITVINOV, E.; ZHAO, F.; ZHENG, T. Electricity Markets in the United States: Power Industry Restructuring Processes for the Present and Future. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2019. v. 17, n. 1, p. 32–42. Available at <<https://doi.org/10.1109/MPE.2018.2872300>>.
- MAGNASCO, A. et al. Data Services for Real Time Optimization of DC Nanogrids with Organic Growth. In: ENERGY PROCEDIA. Online: Elsevier, 2016. v. 103, p. 369–374. ISSN 1876-6102. Available at <<https://doi.org/10.1016/j.egypro.2016.11.301>>.
- MASIELLO, R. Transactive energy: The hot topic in the industry [guest editorial]. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2016. v. 14, n. 3, p. 14–16. Available at <<https://doi.org/10.1109/MPE.2016.2525241>>.
- MEDEIROS, R. Brasil: Modelo de geracão de energia elétrica e políticas públicas para energias incentivadas. In: ASSOCIAÇÃO BRASILEIRA DOS COMERCIALIZADORES DE ENERGIA, 2016, Brasília, DF, Brasil. Slides: ABRACEEL, 2016. Available at <<http://legis.senado.leg.br/sdleg-getter/documento/download/f10aa650-cbf1-420a-b6bb-b305b99df1ae>>. Accessed on March 28, 2018.
- MENGELKAMP, E. et al. Designing microgrid energy markets: A case study: The brooklyn microgrid. In: APPLIED ENERGY. Online: Elsevier, 2018. v. 210, p. 870–880. ISSN 0306-2619. Available at <<https://doi.org/10.1016/j.apenergy.2017.06.054>>.
- \_\_\_\_\_. A blockchain-based smart grid: towards sustainable local energy markets. In: COMPUTER SCIENCE - RESEARCH AND DEVELOPMENT. Online: Springer, 2018. v. 33, n. 1, p. 207–214. ISSN 1865-2042. Special Issue Paper. Available at <<https://doi.org/10.1007/s00450-017-0360-9>>. Accessed on Mai 2, 2018.
- MERZ, M. Potential of the blockchain technology in energy trading. In: \_\_\_\_\_. *Blockchain technology Introduction for business and IT managers*. [S.l.]: De Gruyter Oldenbourg, 2016. cap. 2, p. 51–98. Available at <[http://www.ponton.de/downloads/mm/Potential-of-the-Blockchain-Technology-in-Energy-Trading\\_Merz\\_2016.en.pdf](http://www.ponton.de/downloads/mm/Potential-of-the-Blockchain-Technology-in-Energy-Trading_Merz_2016.en.pdf)>. Accessed on April 11, 2017.
- MIHAYLOV, M. et al. Nrgcoin: Virtual currency for trading of renewable energy in smart grids. In: INTERNATIONAL CONFERENCE ON THE EUROPEAN ENERGY MARKET (EEM14), 11th., 2014, Krakow, Poland. Online: IEEE, 2014. p. 1–6. ISSN 2165-4093. Available at <<https://doi.org/10.1109/EEM.2014.6861213>>. Accessed on Aug 21, 2018.
- MOUGAYAR, W.; BUTERIN, V. *The Business Blockchain*: Promise, Practice, and Application of the Next Internet Technology. [S.l.]: John Wiley & Sons, 2016. 208 p. Ebook. Available at: <<https://books.google.com.br/books?id=CEsPDAAAQBAJ>>. ISBN 978-1-119-30033-5.
- NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Online, 2009. Available at: <<https://bitcoin.org/en/bitcoin-paper>>. Accessed on September 13, 2017.
- NEO. *White Paper*. 2018. Online. Available at: <<https://docs.neo.org/docs/en-us/basic/whitepaper.html>>. Accessed in 2018.

- OLIVEIRA, F. J. A. de. Planejamento da operação do sistema interligado nacional. In: WORKSHOP DA OPERAÇÃO, 1st., 2006, AES-Tietê, Bauru, SP, Brasil. Slides, 2006.
- OLKEN, M. Transactive energy. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2016. v. 14, n. 3, p. 4. Available at <<https://doi.org/10.1109/MPE.2016.2525878>>.
- Operador Nacional do Sistema Elétrico. *O Sistema Interligado Nacional*. 2017. Online. Available at: <<http://www.ons.org.br/paginas/sobre-o-sin/o-que-e-o-sin>>. Accessed in 2017.
- PACITTI, E.; AKBARINIA, R.; EL-DICK, M. *P2P Techniques for Decentralized Applications*. [S.l.]: Morgan & Claypool Publishers, 2012. Ebook. (Synthesis Lectures on Data Management). ISBN 978-1-60845-822-6.
- PECK, M. E. Blockchain: How They Work and Why They'll Change the World. *IEEE Spectrum International*, aug 2017. ISSN 0018-9235. Special Report. Available at: <<https://spectrum.ieee.org/magazine/2017/October>>. Accessed on October 3, 2017.
- PERLMAN, L. *Distributed Ledger Technologies and Financial Inclusion*. [S.l.], 2017. ITU-T Focus Group Digital Financial Services. Available at: <<https://www.itu.int/en/ITU-T/focusgroups/dfs/Pages/default.aspx>>. Accessed on March 20, 2018.
- PETERSON, L. L.; DAVIE, B. S. *Computer Networks*: A systems approach. 5. ed. [S.l.]: Elsevier, 2011. Ebook. ISBN 978-0-12-385059-1.
- POWER LEDGER. *Whitepaper*. [S.l.]: Power Ledger Pty Ltd, 2017. Online. Available at: <<https://www.powerledger.io/>>.
- PricewaterhouseCoopers. *Blockchain: an opportunity for energy producers and consumers?* [S.l.], 2016. Available at: <<https://www.pwc.com/gx/en/industries/energy-utilities-resources/publications/opportunity-for-energy-producers.html>>. Accessed on April 11, 2017.
- R3. *Enterprise Blockchain Platform & Technology*. 2019. Online. Available at: <<https://www.r3.com/>>.
- RADU; ROXANA; ALEIXO, G. *Blockchain e o futuro da Governança*. Rio de Janeiro, RJ, Brasil: Institute for Technology and Society of Rio de Janeiro (ITS Rio), 2015. Online. Available at: <<https://goo.gl/5ZT2Uf>>. Accessed on March 22, 2017.
- SILVA, L. L. da. *The electricity generation sector in Brazil: the perception of regulatory and environmental risk*. Master's Thesis — The George Washington University, Washington, DC, USA, dec 2007. Available at <<http://www.aneel.gov.br/trabalhos-academicos>>. Accessed on April 9, 2018.
- SINGH, J.; MICHELS, J. D. Blockchain As a Service: Providers and trust. Queen Mary School of Law Legal Studies Research, n. 269, dec 2017. Available at: <<https://ssrn.com/abstract=3091223>>. Accessed on March 20, 2018.
- SWAN, M. *Blockchain: Blueprint for a new economy*. [S.l.]: O'Reilly Media, Inc., 2015. Ebook.

- TOFFLER, A. *The Third Wave*. Bantam. [S.l.]: William Morrow, 1980. PDF. (Future Shock). ISBN 0-553-24698-4.
- UK Government Chief Scientific Adviser. *Distributed Ledger Technology: beyond block chain*. Ref: GS/16/1. Government Office for Science, United Kingdom, 2016. 88 p. Available at: <<https://www.gov.uk/government/publications/distributed-ledger-technology-blackett-review>>. Accessed on April 16, 2018.
- United Nations. *The Sustainable Development Goals Report*. New York, NY, USA: Department of Economic and Social Affairs (DESA), 2017. 64 p. Available at: <<https://unstats.un.org/sdgs/files/report/2017/TheSustainableDevelopmentGoalsReport2017.pdf>>. ISSN 2518-3958. ISBN 978-92-1-101368-9.
- VANGULICK, D.; CORNÉLUSSE, B.; ERNST, D. Blockchain for peer-to-peer energy exchanges: design and recommendations. In: POWER SYSTEMS COMPUTATION CONFERENCE (PSCC), 20th., 2018, Dublin, Ireland. *Proceedings*. Online: IEEE, 2018. p. 1–7. ISBN 978-1-910963-10-4. Available at <<http://dx.doi.org/10.23919/PSCC.2018.8443042>>. Accessed on Jan 4, 2019.
- WALKER, G. *Base58*: A set of characters that's easy to share. [S.l.]: Learn me a bitcoin, 2020. Online. Available at: <<https://learnmeabitcoin.com/guide/base58>>. Accessed on Jan 30, 2020.
- WANG, N. et al. Peer-to-Peer Energy Trading among Microgrids with Multidimensional Willingness. *Energies*, Online, n. 12, 2018. ISSN 1996-1073. Available at: <<http://doi.org/10.3390/en11123312>>. Accessed on Jan 4, 2019.
- WIEN ENERGIE. *Wien Energie's new direction: Powering the digital future*. 2016. Online. (Annual Review). Available at: <<https://www.wienenergie.at/eportal3/ep/channelView.do/pageTypeId/67860/channelId/-51877>>. Accessed on August 2, 2018.
- WIKIPEDIA. *Unix time*. [S.l.]: Wikimedia Foundation, Inc., 2019. Online. Available at: <[https://en.wikipedia.org/wiki/Unix\\_time](https://en.wikipedia.org/wiki/Unix_time)>. Accessed on Aug 13, 2019.
- WOOD, G. Ethereum: a secure decentralised generalised transaction ledger. Online, 2016. EIP-150 REVISION. Also known as Yellow Paper. Available at: <<https://gavwood.com/paper.pdf>>. Accessed in 2017.
- World Energy Council. *The developing role of blockchain*. 1. ed. London, United Kingdom, 2017. 88 p. Available at: <<https://www.worldenergy.org/publications>>. Accessed on February 9, 2018. White Paper. In collaboration with PricewaterhouseCoopers (PwC).
- XU, X. et al. The blockchain as a software connector. In: WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE (WICSA), 13th., 2016, Venice, Italy. Online: IEEE, 2016. p. 182–191. ISBN 978-1-5090-2131-4. Available at <<https://doi.org/10.1109/WICSA.2016.21>>. Accessed on April 16, 2018.
- \_\_\_\_\_. A taxonomy of blockchain-based systems for architecture design. In: IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ARCHITECTURE (ICSA), 2017, Gothenburg, Sweden. Online: IEEE, 2017. p. 243–252. ISBN 978-1-5090-5729-0. Available at <<https://doi.org/10.1109/ICSA.2017.83>>. Accessed on April 16, 2018.

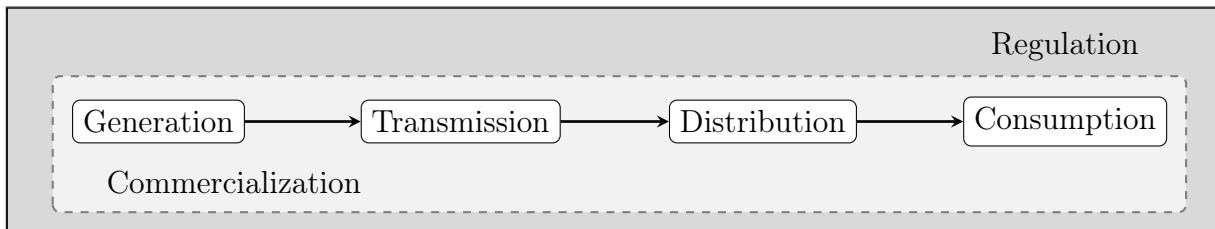
ZHENG, M. et al. Economic and environmental benefits of coordinating dispatch among distributed electricity storage. In: APPLIED ENERGY. Online: Elsevier, 2018. v. 210, p. 842–855. ISSN 0306-2619. Available at <<https://doi.org/10.1016/j.apenergy.2017.07.095>>.

## APPENDIX A – An overview of the electricity sector

Despite the simplicity of the electric nature, the electricity sector is complex in operation and regulation. Nonetheless, when considering the power demand, security to transport and to generate energy, fair investments to keep all the infrastructure working, standards to better integrate markets and industries, laws to guide and protect users and so on, the sector starts to take its format. The number of consumers to count in is not just one more specification, but the main reason why all this complexity exist.

Generally, the electric power infrastructure is presented as the electricity sector behaviour. However, this approach results in a lack of knowledge on regulations and energy trade. The former comprises the energy flow through generation to consumption spots, its equipment and person hour, which has possessing the same structure anywhere. Whilst the latter encompasses the economic, regulatory and political subjects, as shown in Figure 21. The countries have similar approaches to manage electricity, but do so accordingly with its personal scenario.

Figure 21 - The slight difference between the electricity sector concepts.



Legend: While the electric power basic infrastructure is on white boxes, the electricity sector pattern involves greater integrations as shown by gray boxes. The presence and relationships of the agents on each segment is indicated as well.

Source: The author, 2019.

Also, Figure 21 captures the stakeholders and their common relationship, but their classifications varies by means of power level and their roles in the sector. Firstly, the stakeholders responsible to allow the energy flow are denominated as agents, in Brazil, while the stakeholders that financially support the energy market are called consumers. Although both are important to keep the energy market working, they have different rules to conform to based on how they interact with the sector. Secondly, the segment of commercialization represented here is only a broad view of what happens in the energy market but does not represent the restrictions a given agent may have due to specific regulation. For instance, Figure 1 (Chapter 1) details the commercialization in one level, splitting the segment in the two kinds of medium/low voltage power consumer, i.e., those with and without DG.

The consumers always represent the end of the electricity chain, and might be

placed after the distributor or transmitter agents. Therefore, they are essentially denominated as (i) regulated (or captive) consumers, these ones fed by distribution utilities with fixed monthly tariffs; and (ii) free consumers, which are fed by whoever has the capacity to feed its power needs by means of a bilateral agreement (CCEE, 2017b; SILVA, 2007). The former, which is our target group, are subclassified in the following classes: residential, industrial, commercial, rural and public consumers (ANEEL, 2016c).

The possibility of consumers participation in the distribution grid as generators has incentivized the “segment” of Distributed Generation (DG). Together with the increase of ICT management in the electricity sector, which has been evolving towards a smart grid, the DG is one of the predictions with more ruptures in the sector, given its impact in the power infrastructure, and business behaviour (FREITAS; GOULART; GABRICH, 2014).

The following paragraphs present an overview of the Brazilian electricity sector, emphasizing the DG context and some other aspects of the sector that were useful to support the development of the proposed application.

### A.1 The electricity sector in Brazil: a DG approach

Usually, Brazilians can supply their power demands by renewable energy or by natural gas cogeneration systems and “storage” their surplus power in the local distribution grid as power credits for later consumption (ANEEL, 2016a). This power source alternative is known as Distributed Generation (DG) and it categorized as micro/minigrid accordingly with technical requirements of the power generation capacity. Besides individual benefits, the whole sector can get positive impacts, such as “postponement of investments in expansion of transmission and distribution systems, low environmental impact, reduction in network loading, minimization of losses and diversification of the energy matrix” (ANEEL, 2016a).

Despite the appealing invitation to join this manner of power generation, the economic feasibility to allow its expansion is still high for most of the citizens, even with recently government tax incentives (HAHN; MIGHELÃO, 2017) and improvements on technical specifications<sup>30</sup>. For instance, the DG of the solar photovoltaic type had no significant involvement of residential and commercial classes in the national power source, in which both represented only 3 GWh on 2014, i.e., much less than 0.00% of the national generation. On the other hand, for 2023, it is expected a rise up of one-third for both

---

<sup>30</sup> The ANEEL’s document Office Number 720, March 25, 2014 details this information. It can be read in Portuguese at: [aneel.gov.br/arquivos/PDF/Retificação\\_1\\_da\\_Revisão\\_3.pdf](http://aneel.gov.br/arquivos/PDF/Retificação_1_da_Revisão_3.pdf).

classes, which should represent a sum of 1072  $GWh$ , i.e., 0,66% of the national power generation (EPE, 2014b, table 9).

Moreover, the consequences of the DER in the system reliability are always on focus because its particular characteristic of generation within a period implies in new challenges such as: load forecasting; interference on voltage levels; guidances to handle with the amount of information; necessity of specialized command and mechanisms (LEW et al., 2017); optimization of generation in small self-sustaining communities with use of electric vehicles and network balancing (COELHO et al., 2016b) in order to mitigate power quality problems; and availability of active power as demanded by loads (ACKERMANN et al., 2017).

In addition to these aforementioned points, economic issues can arise such as cross-subsidies (IEA, 2011; EPE, 2014b), in which upper tariffs are needed to compensate the utility revenue diminished by the presence of DG in the grid. In summary, it means that general people subsidy the ones benefiting with DG installations. Furthermore, ones also considered the DG a business model disruption that can threaten the economic equilibrium of the power services (GIANELLONI et al., 2017).

On the other hand, the electricity has a huge impact in the national economic growth, so the Government has particular directives to manage and operate the power flow throughout the country. One of them is maintaining the power network as a unit, known as National Power Grid System (SIN), where a set of installations and types of equipment are electrically connected across regions, which is grouped in four subsystems, to allow proper power supply (ONS, 2017). Nowadays, Brazil has in operation almost 160 $GW$  of installed power when considering the sum of hydroelectric plants of any size, which represents more than 60% of the total power consumption, while thermoelectric plants accounting for 26% (ANEEL, 2017a). It supplies electricity for each 209 million of inhabitants (IBGE, 2018). However, only 43.934 consumer units take advantage of DG totalizing a installed power capacity of almost 374 $MW$ , with photovoltaic solar power constituting 77% of it (ANEEL, 2017a).

Furthermore, in the current fundamentals to manage the national electric grid there are three principles: (i) seek the lowest feasible tariff and price; (ii) ensure security in the supply of electricity – by guaranteeing enough generation on reducing high risks notion in this sector and allowing fair return to investors; and (iii) promote social integration – connecting isolated areas to the SIN or, meanwhile, through programs to provide off-grid energy for citizens(CCEE, 2017d; SILVA, 2007).

The regulatory structure that supports this sector is directed by the **Brazilian Ministry of Mines and Energy (MME)**, which, in conjunction with other administrations, aims to better serve the electricity demand by the planning of power generation expansion and attracting required private capital investments. In short, the supporting administrative bodies and their assignments are: **Brazilian Electricity Regulatory**

**Agency (ANEEL)** – responsible for regulating and carrying out long-term investments in the sector, such as develop tariff calculation methodologies for the various segments; **Electric Power Trading Chamber (CCEE)** – responsible for energy trade subjects, such as managing “long-term bilateral contracts among generators and distributions utilities and the settlement of contractual differences for all market agents” (SILVA, 2007, p.11); **Power Sector Monitoring Committee (CMSE)** – which permanently evaluate the security of electricity supply; **Energy Research Company (EPE)** – responsible for planning the long-term electricity sector (10- and 20-year expansion studies); and **National Power System Operator (ONS)** – responsible for operational control and management of the generation and transmission facilities at SIN (CCEE, 2017d; ANEEL, 2016c; SILVA, 2007).

Additionally, the sector operates by two trading markets (supply markets) that comprises the **Regulated Contracting Environment (ACR)** where regulated (or captive) consumers are supplied by distribution utilities under a supervised energy trade managed by ANEEL; and the **Free Contracting Environment (ACL)** where free consumers must negotiate electricity price directly with supply agents (generation or commercialization agents) through freely bilateral contracts signed at CCEE (CCEE, 2017a; SILVA, 2007). Although, ACL could look more tempting, the consumer role is specified by the technical regulations and power conditions management, which limit the transition between one market to another, but guarantees a well planning and operation of the whole grid.

The prevalent trade model emphasizes actions towards customer-centered policies, mainly at the ACR context. In addition, “to avoid the charge of unjustified hidden costs by distribution companies for energy supplied to captive consumers” (SILVA, 2007, p.11), the purchase of power by distribution agents must be through auctions carried out by CCEE, on behalf of ANEEL, which, in turn, uses the criterion of lowest price of generation in order to reduce the acquisition cost of electricity to be passed for captive consumers (CCEE, 2017d).

## Finance matters

The electricity tariff aims to ensure sufficient revenue for distribution utilities to cover operating costs, to remunerate necessary investments for the expansion of power capacity, to guarantee quality services, and to create incentives for efficiency (ANEEL, 2017c; ANEEL, 2016c).

As previously said, in the ACR market – which DG lands on – customers are still classified by classes such as residential, industrial, commercial, rural and public energy. But, whilst all of them are captive consumers and have the lowest possible energy price

offer supervised by ANEEL, they differ in terms of tariff granularity by levels of power consumption and demand. Small power consumers pay only for the energy consumption, while greater consumers pay for consumption and demand, and so, they also have the condition to opt for tariff diversification by time frames.

The electricity tariff is charged by energy consumed (R\$/kWh) and availability, 24 hours a day throughout the whole year. It is strictly regulated by ANEEL, because it is an essential good, and it is composed of costs incurred from generation and transmission segments, which indeed is the remarkable component that represents more than half of the final value (ANEEL, 2017c).

The cost of the generation consists of each power plant availability present in the SIN (CCEE, 2017c), and forecasting analysis of the hydroelectric generation feasibility at the present time and in the future by periods and subsystems (OLIVEIRA, 2006). This calculation made by ONS also consider the power transmission capacity between each subsystem and the energy availability in each time frame for defining the best price (CCEE, 2017c).

Thereby, the final generation cost definition stands between its lowest level, just given by hydroelectric costs, to the highest level, just operated by thermoelectric ones (OLIVEIRA, 2006). The former is not only due to the quantity offered, but mainly due to the efficiency when compared to the cost of installation and the “fuel” used (CCEE, 2017c). Nonetheless, the latter is very worth, because they can be dispatched anytime without restrictions imposed by water limits. Notably, the former is preferred to keep grid powered on, whilst the latter is used to keep system working well during power peaks. The optimization of these costs guarantees the modality of tariffs in all subsystems of the SIN and uninterrupted electricity for all Brazilian citizens (OLIVEIRA, 2006).

Although the above methodology is a public statement, it has started to be clearly available to the population only in 2015 (ANEEL, 2017b), with the value of electricity being detailed in the energy bill. In other words, since that year the monthly cost of power generation has started to be visually discriminated in the energy bill, by showing a flag that can have three different colours – green, yellow or red – to indicate generation threshold. A determination that aims to educate residential consumers about power generation conditions in the country, i.e., the Tariff Flags System depicts unfavourable situations in the SIN, where there is a need for a thermoelectric generation to compensate (or to ensure future) hydroelectric generation (ANEEL, 2017b).

Another measurement recently announced to be extended to the residential class is the tariff diversification according to the day and time of power consumption. In other words, during working days the period of high demand of power, known as peak time, has higher tariff, otherwise the value is lower than the conventional tariff (ANEEL, 2017d).

As can be seen, the DG can come to the rescue of diversified price options for the power generation. Faced with the range of possibilities from the different types of power

source (single-wind turbines, biomass generators, solar power and so on), to equipment technology, and local of installation (rural or urban), consumers may benefit from a personal alternative tariff based on the pros and cons in sharing with others the financing of a DG project (ANEEL, 2016a).

## APPENDIX B – The terms used throughout the DApp development

<b>Person</b>	an independent agent which interacts with the blockchain ledger.
<b>Reader</b>	a person with access to only read the blockchain public information (eg.: a power utility representative).
<b>User</b>	a person with an account in the blockchain platform, i.e., she/he has a pair of keys (public and private) and the permission to write in the blockchain.
<b>Member</b>	a user that has gained access from a group to interact with a particular environment of the blockchain platform.
<b>Group</b>	several members united under the Brazilian DG legislation which shares a particular blockchain environment to exchange energy assets.
<b>Caller or invoker</b>	a user whose interacts with the group's smart-contract.
<b>Power</b>	the capacity to generate electricity (measured in watt [ $W$ ]).
<b>Energy</b>	the amount of power during a given time of 1 hour (measured in watt-hour [ $Wh$ ]).
<b>Quota</b>	the share value a member has in relation to the total power generation of the group (represented in percentage [%]).
<b>Token</b>	the digital currency (crypto-currency) of the group used to exchange quotas (valued in <i>Sharing Electricity in Brazil (SEB)</i> ).
<b>Fee</b>	the value to pay for a transaction in the blockchain (usually in terms of specific crypto-currency).
<b>Expense</b>	the amount spent with administrative subjects (may be crypto- or fiat money).
<b>Cost</b>	the amount spent to finance a service/product of the group's interest (in fiat money only).
<b>Transaction</b>	any operation that happens in the blockchain, usually when dealing with the smart contracts.

<b>Exchange</b>	the act of buy/sell/donate quotas regardless of the currency used – tokens (from a member) or fiat-currency (from a new power plant funding) <sup>31</sup> .
<b>Membership</b>	anything related to the user qualification as a member.
<b>Process</b>	a series of operations performed in the treatment of the group's resolution.
<b>Ballot</b>	a group voting process to determine something.
<b>Change</b>	a specific process which deals with the variety of group's registering data.
<b>Smart-contract</b>	a series of functions and logic operations written in some human-readable computer programming language that supports the development of DApps.
<b>Address</b>	the public identification of both smart-contracts and users through a hash number, usually a 17 bytes long
<b>Public key</b>	the public identification of a user through a hash number, usually a 33 bytes long.
<b>Private key</b>	the private identification of a user through a hash number, usually a 32 bytes long.
<b>Consensus</b>	the blockchain method to agree or not about new data to be appended in the ledger which usually follows a random algorithm to coordinate the decision automatically.
<b>Private storage space (persistent space)</b>	a restricted storage area of a blockchain platform that belongs to a smart contract. The implementation of this feature varies by blockchain but the data itself is never appended on the ledger, only the hash that represents it.
<b>Referendum</b>	a voting process to define about something related to the group's asset, indeed, it is the group consensus.
<b>Waiting period</b>	a period during which a process takes place or is projected to occur, normally a wait time for voting or to implement a new power plant.

---

<sup>31</sup> Although the blockchain financing can serve as legal proof that a real currency amount will be traded, the payment method will occur off-chain and should follow the group's own rules to handle with whatever concerns it.

## APPENDIX C – The UML class diagram

The Unified Modeling Language (UML) diagram does not show what to do first and next or how to design the system but it helps to visualize the structure and the communication between objects. It is on the problem-side domain, which means it is used to get a general view of the system to be developed through the understanding of the objects concepts and properties.

In the present scenario, there are four agents identifiable across the Microgrid Transactive Energy smart contract (MTEsm) and one agent under the blockchain system. The former has the three access type to the available MTEsm information – the general person, the general user and the member – plus the group consensus. The later has only the blockchain consensus responsible for what will be appended or not in the ledger.

The comprehension about how the specifications of the aforementioned items relate to each other and compose the smart contract is designed at the UML class diagram presented in Figure 22. The dashed lines highlight different aspects of the DApp accordingly to the following:

- A** The core concepts of blockchain technology are its distributed database (ledger) and its consensus mechanism.
- B** The distinguish approach of an ordinary person (reader) to a general user. While the former can only read the public ledger, the latter has more attributes to interact with the blockchain network. The former becomes the latter through the blockchain registration platform.
- C** Similarly happens between the general user and the member of a group. Nonetheless, the registration process is defined by a group referendum. Now, after the group approval, the latter has access to the group's private space and personalized smart contract operations.
- D** It is the MTEsm environment. Different sets of operations are identifiable, from methods to update a member registering data until energy transaction and management. Some attributes are split to show how similar aspects are presented on different steps of the system, such as the waiting period.

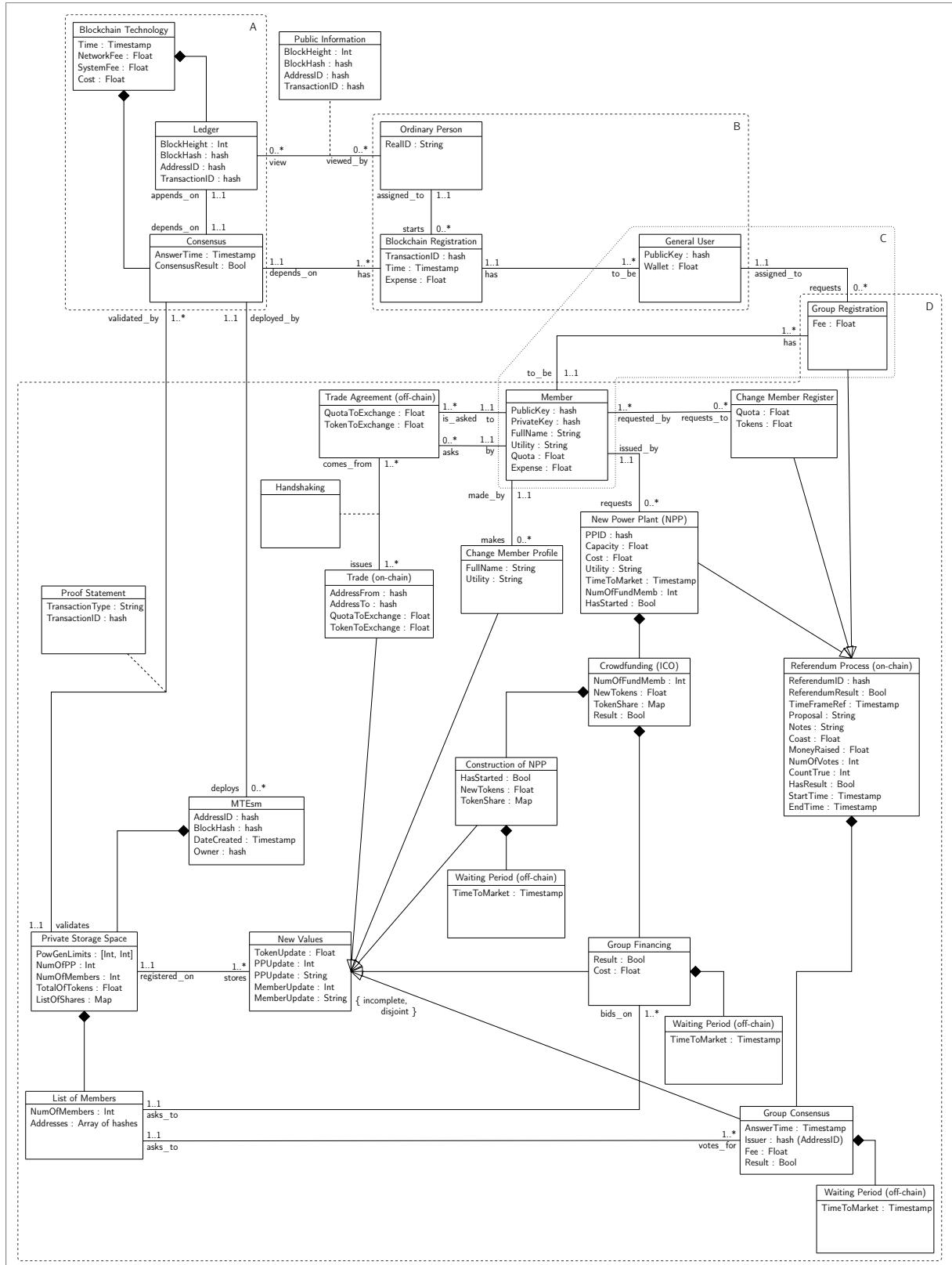
Moreover, the associations guide either singular and plural relationships between agents, and how entities relate to each other. However, only the most relevant relationships between agents and entities are considered, as described below:

- Although *ordinary person*, *general user* and *member* could be defined as a specialization of one another, it was preferred to display the associations with two

consensus methods because this better shows how the access to the ledger information changes. Therefore, it also highlights that the real ID attribute of a person can only be uncovered by someone with access to the MTEsm environment.

- The *Referendum Process* is a generalisation of any request to change a value in the MTEsm private space. It always has a unique identifier for every request and waits for the group consensus to take further action.
- The relationships are not exhaustive and only some of them are presented. For instance, the registering data of power plants and members have quite similar attributes, so the method to update one might work to the other, and hence, only one kind of this relation was considered on the diagram. The notation *incomplete* close to *New Values* denotes these abstractions.
- The associations with external interfaces (off-chain) are easily identifiable too. The way any trading process must proceed is detached from the check-out step that formalizes the energy trade (on-chain). And the waiting period required for a given operation to happen is counted away of how the blockchain performs the analysis to allow or not it.
- The composite aggregation (black diamonds) states the unique relationship between objects. For example, each referendum process has only one group consensus, or even each crowdfunding process is solely related to the construction of a new power plant, that represents the unique way to create new SEBs.

Figure 22 - The UML class diagram of the DApp proposed.



Source: The author, 2019.

## APPENDIX D – The descriptions of the considered constraints

To simplify the flowcharts presented in Section 3.3.1, most of the constraints were reduced to only one decision box to represent several validation steps. However, to keep the documentation about them, the following tables connect the functions' constraints with the failure message the MTEsm should return to the user.

Table 8 - The description of the constraints identified in the MTEsm (to be continued).

constraints	failure message
<b>Were all the required inputs provided?</b>	Please provide the 2 arguments: your full name, and the power utility name.
<b>Is the invoker requesting the admission for her-/himself?</b>	The admission can not be done on someone else's behalf.
<b>Has the invoker joined the group yet?</b>	Thanks, you're already a member. We're glad to have you as part of the group!
<b>Was the referendum ID provided?</b>	Please provide only the admission process ID.
<b>Did the time frame pass out?</b>	There isn't a result yet.

(a) For ADMISSION (Figure 7, Section 3.3.1).

constraints	failure message
<b>Were all the required inputs provided?</b>	Provide at least a member address or a PP ID.

(b) For SUMMARY (Figures 8 to 9, Section 3.3.1).

constraints	failure message
<b>Were all the required inputs provided?</b>	Please provide the 2 arguments: the referendum ID, and your vote.
<b>Is the invoker voting for her-/himself?</b>	The vote can not be done on someone else's behalf.
<b>Still time to vote?</b>	The ballot has ended.

(c) For VOTE (Figure 10, Section 3.3.1).

Source: The author, 2019.



Table 10 - The description of the constraints identified in the MTEsm (conclusion).

constraints	failure message
<b>Were all the required inputs provided?</b>	Please provide the 4 arguments: the PP capacity, the cost to build it up, the power utility name in which the PP will be installed, and the period to wait the new PP gets ready to operate.
<b>Is it defined a factual time to market?</b>	The time to market must be a factual period.
<b>Does it have one input?</b>	Please provide at least the new PP process ID.
<b>Does it have more inputs than the required?</b>	Please provide at most the new PP process ID, and the PP ID itself if any.
<b>Did the crowdfunding finish?</b>	There isn't a result about the new PP crowdfunding yet.
<b>Did the waiting period of the referendum pass out?</b>	There isn't a result about the new PP request yet.
<b>Did the waiting period of the time to market pass out?</b>	The new PP is not ready to operate yet.

(a) For POWER-UP (Figures 16 to 18, Section 3.3.1).

constraints	failure message
<b>Were all the required inputs provided?</b>	Please provide the 3 arguments: the address of who you are transacting to, the quota value, and the amount of tokens.
<b>Does the first address provided belong to the invoker?</b>	Only the owner of an account can exchange her/his asset.
<b>Is the exchange address also a member?</b>	The address you are transacting to must be a member too.
<b>Do both addresses belong to the same power utility?</b>	Both members must belong to the same power utility cover area.
<b>Are there provided factual values for trade?</b>	You're doing it wrong. To donate energy let ONLY the 3rd argument empty. Otherwise, to donate tokens let ONLY the 2nd argument empty.

(b) For TRADE (Figure 19, Section 3.3.1).

Source: The author, 2019.

## APPENDIX E – The MTEsm

The MTEsm code below refers to the version 1.0 of the development process. Figure 23 summarizes and identifies each slice of the code splitted for better visualization, which follows Figure 5 structure (Section 3.3). The code is also available for discussions beyond the present document at the open source repository: [github.com/yurigabrich/microgrid-dapp/releases/tag/v1.0](https://github.com/yurigabrich/microgrid-dapp/releases/tag/v1.0).

Figure 23 - The code folding and cross-referencing of the MTEsm.

```

1   Code E.1    // libraries
...
8   namespace Neo.SmartContract
9   {
10      public class MTEsm : Framework.SmartContract
11      {
12      //-----
13      // EVENTS
Code E.2
...
132     // GLOBAL VARIABLES
Code E.3, Code E.4 and Code E.5
...
132     // THE MAIN INTERFACE
Code E.6, Code E.7, Code E.8, Code E.9 and Code E.10
...
387     // GROUP FUNCTIONS – The restrictions are made in the ‘Main’.
Code E.11, Code E.12, Code E.13, Code E.14, Code E.15 and Code E.16
...
634     // ADMINISTRATIVE FUNCTIONS
Code E.17, Code E.18, Code E.19, Code E.20, Code E.21 and Code E.22
...
911     // SYSTEM FUNCTIONS
Code E.23, Code E.24, Code E.25, Code E.26 and Code E.27
...
1136    // METHODS FOR MEMBERS
Code E.29, Code E.30 and Code E.31
...
1253    // METHODS FOR POWER PLANTS
Code E.32, Code E.33 and Code E.34
...
1398    // METHODS FOR REFERENDUMS
Code E.35 and Code E.36
...
1500    // METHODS TO FINANCE A NEW POWER PLANT
Code E.37 and Code E.38
...
1611    }
1612 }
```

Source: The author, 2019.

## Code E.1 - Libraries.

```

1 using Neo.SmartContract.Framework;
2 using Neo.SmartContract.Framework.Services.Neo;
3 using Neo.SmartContract.Framework.Services.System;
4 using System;
5 using System.ComponentModel;
6 using System.Numerics;

```

## Code E.2 - EVENTS.

```

15 [DisplayName("transaction")]
16 public static event Action<byte[], byte[], BigInteger, BigInteger> Transfer;
17 [DisplayName("transaction")]
18 public static event Action<string, byte[], BigInteger, BigInteger> Retract;
19 [DisplayName("membership")]
20 public static event Action<byte[], string> Membership;
21 [DisplayName("process")]
22 public static event Action<string, string> Process;
23 [DisplayName("ballot")]
24 public static event Action<string, byte[], bool> Ballot;
25 [DisplayName("offer")]
26 public static event Action<string, byte[], BigInteger> Offer;
27 [DisplayName("change")]
28 public static event Action<string, object> Update;
29 [DisplayName("invalid operation")]
30 public static event Action<string> Exception;

```

## Code E.3 - GLOBAL VARIABLES (to be continued).

```

35 // The total number of referendum processes.
36 private static BigInteger NumOfRef() => Storage.Get("numoref").AsBigInteger();
37
38 // The total number of power plant (PP) units.
39 private static BigInteger NumOfPP() => Storage.Get("numofpp").AsBigInteger();
40
41 // The total number of members.
42 private static BigInteger NumOfMemb() => Storage.Get("numofmemb").AsBigInteger();
43
44 // The group total power supply, i.e., sum of PP's capacity.
45 private static BigInteger TotalSupply() => Storage.Get("totalsupply").
46     AsBigInteger();
47
48 // The member's dataset settings.
49 private static string[] profile => new string[] {"fullname", "utility"};
50 private static string[] register => new string[] {"quota", "tokens"};
51 private struct MemberData
52 {
53     public static StorageMap ID => Storage.CurrentContext.CreateMap(nameof(ID));
54     public static StorageMap FullName => Storage.CurrentContext.CreateMap(nameof(
55         FullName));
56     public static StorageMap Utility => Storage.CurrentContext.CreateMap(nameof(
57         Utility));
58     public static StorageMap Quota => Storage.CurrentContext.CreateMap(nameof(Quota
59         ));
60     public static StorageMap Tokens => Storage.CurrentContext.CreateMap(nameof(
61         Tokens));
62 }

```

## Code E.4 - GLOBAL VARIABLES (continuity).

```

59     // The referendum's dataset settings.
60     private struct RefData
61     {
62         public static StorageMap ID => Storage.CurrentContext.CreateMap(nameof(ID));
63         public static StorageMap Proposal => Storage.CurrentContext.CreateMap(nameof(
64             ↪ Proposal));
65         public static StorageMap Notes => Storage.CurrentContext.CreateMap(nameof(Notes
66             ↪ ));
67         public static StorageMap Cost => Storage.CurrentContext.CreateMap(nameof(Cost))
68             ↪ ;
69         public static StorageMap Address => Storage.CurrentContext.CreateMap(nameof(
70             ↪ Address));
71         public static StorageMap Time => Storage.CurrentContext.CreateMap(nameof(Time))
72             ↪ ;
73         public static StorageMap MoneyRaised => Storage.CurrentContext.CreateMap(nameof(
74             ↪ (MoneyRaised)));
75         public static StorageMap NumOfVotes => Storage.CurrentContext.CreateMap(nameof(
76             ↪ NumOfVotes));
77         public static StorageMap CountTrue => Storage.CurrentContext.CreateMap(nameof(
78             ↪ CountTrue));
79         public static StorageMap Outcome => Storage.CurrentContext.CreateMap(nameof(
80             ↪ Outcome));
81         public static StorageMap HasResult => Storage.CurrentContext.CreateMap(nameof(
82             ↪ HasResult));
83         public static StorageMap StartTime => Storage.CurrentContext.CreateMap(nameof(
84             ↪ StartTime));
85         public static StorageMap EndTime => Storage.CurrentContext.CreateMap(nameof(
86             ↪ EndTime));
87     }
88
89     // The PP's dataset settings.
90     private struct PPData
91     {
92         public static StorageMap ID => Storage.CurrentContext.CreateMap(nameof(ID));
93         public static StorageMap Capacity => Storage.CurrentContext.CreateMap(nameof(
94             ↪ Capacity));
95         public static StorageMap Cost => Storage.CurrentContext.CreateMap(nameof(Cost))
96             ↪ ;
97         public static StorageMap Utility => Storage.CurrentContext.CreateMap(nameof(
98             ↪ Utility));
99         public static StorageMap TimeToMarket => Storage.CurrentContext.CreateMap(
100             ↪ nameof(TimeToMarket));
101         public static StorageMap NumOfFundMemb => Storage.CurrentContext.CreateMap(
102             ↪ nameof(NumOfFundMemb));
103         public static StorageMap HasStarted => Storage.CurrentContext.CreateMap(nameof(
104             ↪ HasStarted));
105     }

```

## Code E.5 - GLOBAL VARIABLES (conclusion).

```

89     // The ICO's dataset settings (for crowdfunding).
90     private struct ICOData
91     {
92         public static StorageMap StartTime => Storage.CurrentContext.CreateMap(nameof(
93             ↵ StartTime));
94         public static StorageMap EndTime => Storage.CurrentContext.CreateMap(nameof(
95             ↵ EndTime));
96         public static StorageMap TotalAmount => Storage.CurrentContext.CreateMap(nameof(
97             ↵ (TotalAmount)));
98         public static StorageMap Contributions => Storage.CurrentContext.CreateMap(
99             ↵ nameof(Contributions));
100        public static StorageMap Success => Storage.CurrentContext.CreateMap(nameof(
101            ↵ Success));
102        public static StorageMap HasResult => Storage.CurrentContext.CreateMap(nameof(
103            ↵ HasResult));
104
105        public static StorageMap Bid => Storage.CurrentContext.CreateMap(nameof(Bid));
106    }
107
108    // The predefined periods to answer both a referendum and a crowdfunding, and to
109    // wait until a PP construction.
110    private const uint timeFrameRef = 259200;           // 30 days
111    private const uint timeFrameCrowd = 518400;          // 60 days
112    private const uint minTimeToMarket = 259200;         // 30 days
113
114    // The essential settings to support the process of a new PP crowdfunding.
115    private const int minOffer = 100;                  // Brazilian Reais (R$)
116    private const uint factor = 1000;                 // 1kW == 1SEB
117
118    // The token basic settings.
119    private static string Name() => "Sharing Electricity in Brazil";
120    private static string Symbol() => "SEB";
121
122    // The power limits of the distributed generation category defined by Brazilian
123    // law (from 0MW to 5MW).
124    private static int[] PowGenLimits() => new int[] {0, 5000000};
125
126    // The time a given function is invoked.
127    private static uint InvokedTime() => Blockchain.GetHeader(Blockchain.GetHeight())
128        .Timestamp;
129
130    // The trick to lock the admission operation process without a referendum.
131    private static void OnlyOnce() => Storage.Put("firstcall", 1);
132
133    // The trick to support the conversion from 'int' to 'string'.
134    private static string[] Digits() => new string[10] {"0", "1", "2", "3", "4", "5",
135        ↵ "6", "7", "8", "9"};
136
137    // The characters of the Base58 scheme.
138    private const string Alphabet = "123456789
139        ↵ ABCDEFGHJKLMNPQRSTU VWXYZabcdefghijklmnopqrstuvwxyz";

```

Code E.6 - THE MAIN INTERFACE (to be continued).

```

134  public static object Main ( byte[] address , string operation , params object []
135    ↵ args )
136    {
137      // General operation .
138      if ( operation == "admission" )
139      {
140        if ( args.Length != 2 )
141          return Warning("Please provide the 2 arguments: your full name, and the
142            ↵ power utility name .");
143
144        if ( ( (string)GetMemb(address) ).Length != 0 )
145          return Warning("The admission can not be done on someone else 's behalf .");
146
147        if ( Storage.Get("firstcall").AsBigInteger() == 0 )
148        {
149          // No admission process is required .
150
151          // Locks this 'if' statement .
152          OnlyOnce();
153
154          // Defines the 'invoker/caller' as the first member .
155          Membership( address , "Welcome on board ! " );
156          Member( address , (string)args[0] , (string)args[1] , 0 , 0 );
157          return true;
158        }
159
160        return Admission( address ,           // invoker/caller address
161                          (string)args[0] ,   // fullName
162                          (string)args[1] ); // utility
163      }
164
165      // Partially restricted operation .
166      if ( operation == "summary" )
167      {
168        if ( args.Length < 1 )
169          return Warning("Provide at least a member address or a PP ID .");
170
171        if ( ((string)GetMemb((byte[])args[0])).Length != 0 )
172        {
173          // The args[0] is a member , i.e. ,
174          // it has being requested information about a member .
175
176          if ( !Runtime.CheckWitness(address) )
177            return Warning("This request can not be done on someone else 's behalf .");
178
179          if ( ((string)GetMemb(address)).Length == 0 )
180            return Warning();
181        }
182
183        return Summary( (object)args[0] , // any ID
184                      (string)args[1] ); // desired option
185      }
186

```

## Code E.7 - THE MAIN INTERFACE (continuity).

```

188     // Restricted operations.
189     if ( ((string)GetMemb(address)).Length != 0 )
190     {
191
192         // Group operations.
193         if ( operation == "vote" )
194         {
195             if ( args.Length != 2 )
196                 return Warning("Please provide the 2 arguments: the referendum ID, and
197                 ↪ your vote.");
198
199             if ( !Runtime.CheckWitness(address) )
200                 return Warning("The vote can not be done on someone else's behalf.");
201
202             if ( isLock( (string)args[0]) )
203                 return Warning("The ballot has ended.");
204
205             return Vote( (string)args[0],      // referendum ID
206                         address,           // member address
207                         (bool)args[1] );   // vote answer
208         }
209
210         if ( operation == "bid" )
211         {
212             if ( args.Length != 2 )
213                 return Warning("Please provide the 2 arguments: the PP ID, and your bid."
214                 ↪ );
215
216             if ( !Runtime.CheckWitness(address) )
217                 return Warning("The bid can not be done on someone else's behalf.");
218
219             if ( (((string)args[0])[0] != 'P') || (((string)args[0]).Length == 0) )
220                 return Warning("Provide a valid PP ID.");
221
222             if ( (GetPP((string)args[0], "utility")) != (GetMemb(address, "utility")) )
223                 return Warning("This member cannot profit from this power utility.");
224
225             if ( (int)args[1] <= minOffer )
226                 return Warning(String.Concat("The minimum bid allowed is R$ ", Int2Str(
227                 ↪ minOffer)));
228
229             if ( isLock( (string)args[0] ) )
230                 return Warning("The crowdfunding has ended.");
231
232             return Bid( (string)args[0],      // PP ID
233                         address,           // member address
234                         (BigInteger)args[1] ); // bid value
235         }
236
237         if ( operation == "change" )
238         {
239             if ( args.Length != 2 )
240                 return Warning("Please provide 2 arguments only. The first one must be
241                 ↪ either the identification of the member (address) or the PP (ID). The second
242                 ↪ one must be an array. It can be either the options about the data that will be
243                 ↪ changed, or an empty array to request the deletion of something.");

```

Code E.8 - THE MAIN INTERFACE (continuity).

```

239         // To simplify the indexing.
240         var opt = (object[]) args[1];
241
242         // Should be a PP ID.
243         if ( IsValidId(args[0]) )
244         {
245             if ( ((string)GetPP((string)args[0], "utility")).Length == 0 )
246                 return Warning("Provide a valid PP ID.");
247
248             if ( opt.Length != 1 )
249                 return Warning("Only one option is required to update a PP subject. It
250     ↪ can be a PP utility name, or a new bid value for a PP crowdfunding campaign.");
251
252             if ( opt.Length == 1 )
253                 if ( isLock( (string)args[0] ) )
254                     return Warning("The crowdfunding has ended.");
255
256             if ( !(Runtime.CheckWitness(address)) )
257                 return Warning("Only the member can change its bid.");
258
259             // Updates the option array to pass the 'address' together with the bid
260     ↪ value.
261             int i = opt.Length;
262             object[] option = new object[i+1];
263
264             while( i > 0 )
265             {
266                 option[i] = opt[i-1];
267                 i--;
268             }
269             option[i] = address;
270
271             return Change( (object)args[0], // PP ID
272                           option );      // array with desired values
273         }
274     }
275
276     // Should be a member ID (address).
277     else
278     {
279         if ( ((string)GetMemb((byte[]) args[0])).Length == 0 )
280             return Warning("Provide a valid member address.");
281
282         if ( (opt.Length != 2) || (opt.Length != 0) )
283             return Warning("Provide valid arguments to update/delete an address.");
284
285         if ( ( ((string)opt[0] == profile[0]) | ((string)opt[0] == profile[1]) )
286     ↪ & !(Runtime.CheckWitness(address)) )
287             return Warning("Only the member can change her/his profile data.");
288     }
289
290     return Change( (object)args[0], // member address or PP ID
291                   opt );          // array with desired values
292 }
```

## Code E.9 - THE MAIN INTERFACE (continuity).

```

293     if ( operation == "power up" )
294     {
295         if ( args.Length != 4 )
296             return Warning("Please provide the 4 arguments: the PP capacity , the cost
297             ↵ to build it up, the power utility name in which the PP will be connected to ,
298             ↵ and the period to wait until the new PP gets ready to operate.");
299
300         return PowerUp( (int)args[0] ,           // capacity [MW]
301                         (int)args[1] ,           // cost [R$]
302                         (string)args[2] ,        // power utility name
303                         (uint)args[3] );        // time to market
304     }
305
306
307     if ( operation == "trade" )
308     {
309         if ( args.Length != 3 )
310             return Warning("Please provide the 3 arguments: the address of who you
311             ↵ are transacting to , the quota value , and the amount of tokens.");
312
313         if ( !Runtime.CheckWitness(address) )
314             return Warning("Only the owner of an account can exchange her/his asset."
315             ↵ );
316
317         if ( ((string)GetMemb((byte[])args[0])).Length == 0 )
318             return Warning("The address you are transacting to must be a member too."
319             ↵ );
320
321         if ( (GetMemb(address , "utility")) != (GetMemb((byte[])args[0] , "utility"))
322             ↵ )
323             return Warning("Both members must belong to the same power utility
324             ↵ coverage area .");
325
326         if ( ((int)args[1] <= 0) & ((int)args[2] <= 0) )
327             return Warning("You're doing it wrong. To donate energy let ONLY the 3rd
328             ↵ argument empty. Otherwise , to donate tokens let ONLY the 2nd argument empty.")
329             ↵ ;
330
331         return Trade( address ,           // from address
332                         (byte[])args[0] ,           // to address
333                         (BigInteger)args[1] ,        // quota exchange
334                         (BigInteger)args[2] ); // token price
335     }
336
337     // Administrative operations .
338     if ( operation == "admission result" )
339     {
340         if ( args.Length != 1 )
341             return Warning("Please provide only the admission process ID .");
342
343         if ( isLock( (string)args[0] ) )
344             return Warning("There isn 't a result yet .");
345
346         return AdmissionResult( (string)args[0] ); // Referendum ID
347     }

```

## Code E.10 - THE MAIN INTERFACE (conclusion).

```

342     if ( operation == "change result" )
343     {
344         if ( args.Length != 1 )
345             return Warning("Please provide only the change process ID.");
346
347         if ( isLock( (string)args[0] ) )
348             return Warning("There isn't a result yet.");
349
350         ChangeResult( (string)args[0] ); // Referendum ID
351     }
352
353     if ( operation == "power up result" )
354     {
355         if ( args.Length == 0 )
356             return Warning("Please provide at least the new PP process ID.");
357
358         if ( args.Length > 2 )
359             return Warning("Please provide at most the new PP process ID, and the PP
360             ↪ ID itself if any.");
361
362         PowerUpResult( (string)args[0], // Referendum ID
363                         (string)args[1] ); // PP ID
364     }
365
366     if ( operation == "list of power plants" )
367     {
368         if ( args.Length != 0 )
369             return Warning("This function does not need attributes.");
370
371         ListOfPPs();
372     }
373
374     if ( operation == "list of members" )
375     {
376         if ( args.Length != 0 )
377             return Warning("This function does not need attributes.");
378
379         ListOfMembers();
380     }
381
382     return Warning("No operation found. Have you wrote it right?");
383 }
```

## Code E.11 - GROUP FUNCTIONS (to be continued).

```

389     // To request to join the group.
390     private static string Admission( byte[] address, string fullName, string utility
391                                     ↪ )
392     {
393         string rID = Ref( fullName, utility, address );
394         Membership( address, "Request for admission." );
395         return rID;
396     }
```

## Code E.12 - GROUP FUNCTIONS (continuity).

```

397     // To get information about something.
398     private static object Summary( object id, string opt = "" )
399     {
400         // If 'id' is a 'byte[]' == member.
401         if ( !IsValidId(id) )
402         {
403             var address = (byte[]) id;
404
405             if ( (opt == "") || (opt == "detailed") )
406             {
407                 object[] brief = new object[] { GetMemb(address), GetMemb(address, "utility" )
408                 ← ), GetMemb(address, "quota"), GetMemb(address, "tokens") };
409
410                 if ( opt == "detailed" )
411                 {
412                     ShowContributedValues( address, ListOfPPs() );
413                 }
414                 return brief;
415             }
416             return GetMemb(address, opt);
417         }
418
419         // If 'id' is a 'string' with prefix 'P' == power plant.
420         else if ( ((string)id)[0] == 'P' )
421         {
422             var ppID = (string)id;
423
424             // The PP's crowdfunding had succeed and the PP is operating.
425             if ( (bool)GetPP(ppID) )
426             {
427                 if ( (opt == "") || (opt == "detailed") )
428                 {
429                     object[] brief = new object[] { GetPP(ppID, "capacity"), GetPP(ppID, "cost" )
430                     ← ), GetPP(ppID, "utility"), GetPP(ppID, "numoffundmemb") };
431
432                     if ( opt == "detailed" )
433                     {
434                         ShowContributedValues( ppID, ListOfMembers() );
435                     }
436                     return brief;
437                 }
438                 return GetPP(ppID, opt);
439             }
440
441             // The PP's crowdfunding may be succeed or not, and the PP is definitely not
442             ← operating.
443             else
444             {
445                 if ( (opt == "") || (opt == "detailed") )
446                 {
447                     object[] brief = new object[] { GetCrowd(ppID, "starttime"), GetCrowd(ppID
448                     ← , "endtime"), GetCrowd(ppID, "totalamount"), GetCrowd(ppID, "contributions"),
449                     ← GetCrowd(ppID, "success") };

```

## Code E.13 - GROUP FUNCTIONS (continuity).

```

446         if ( opt == "detailed" )
447         {
448             foreach ( byte[] member in ListOfMembers() )
449             {
450                 BigInteger bid = GetBid(ppID, member);
451
452                 if ( bid != 0 )
453                 {
454                     Runtime.Notify( new object[] { member, bid } );
455                 }
456             }
457         }
458         return brief;
459     }
460     return GetCrowd(ppID, opt);
461 }
462 }
463
464 // If 'id' is a 'string' with prefix 'R' == referendum process.
465 else if ( ((string)id)[0] == 'R' )
466 {
467     var rID = (string)id;
468
469     if ( opt == "" )
470     {
471         return new object[] { GetRef(rID, "proposal") , GetRef(rID, "notes") , GetRef(
472             rID, "cost") , GetRef(rID, "outcome") };
473     }
474     return GetRef(rID, opt);
475 }
476
477 // Wrap-up the group information.
478 else
479 {
480     return new object[] { PowGenLimits()[0] , PowGenLimits()[1] , NumOfPP(),
481             NumOfMemb() , Name() , Symbol() , TotalSupply() };
482 }
483
484 // To vote in a given process.
485 private static bool Vote( string rID, byte[] member, bool answer )
486 {
487     // Increases the number of votes.
488     BigInteger temp = (BigInteger)GetRef(rID, "numofvotes");
489     UpRef(rID, "numofvotes", temp+1);
490
491     if ( answer )
492     {
493         // Increases the number of "trues".
494         temp = (BigInteger)GetRef(rID, "counttrue");
495         UpRef(rID, "counttrue", temp+1);
496     }
497
498     // Publishes the vote.
499     Ballot(rID, member, answer);
500     return true;
501 }
```

## Code E.14 - GROUP FUNCTIONS (continuity).

```

502  // To make a bid in a new PP crowdfunding process.
503  private static bool Bid( string ppID, byte[] member, BigInteger bid )
504  {
505      BigInteger target = (BigInteger)GetPP(ppID, "cost");
506      BigInteger funds = (BigInteger)GetCrowd(ppID, "totalamount");
507
508      if ( bid > (target - funds) )
509          return Warning( String.Concat(String.Concat("You offered more than the amount
510  ↪ available (R$ ", Int2Str((int)(target - funds)) ), ",00). Bid again!" ));
```

511     // WARNING!

512     // All the following steps are part of a crowdfunding process.

513     // Although the PP already has a register (i.e. a PP ID),

514     // it does not have started to operate (PPData.HasStarted = false).

515

516     // Increases the value gathered so far.

517     UpCrowd(ppID, "totalamount", funds + bid);

518

519     // Increases the number of contributions.

520      temp = (**BigInteger**)GetCrowd(ppID, "contributions");

521     UpCrowd(ppID, "contributions", temp+1);

522

523     // Tracks bid by member for each ICO process.

524     UpBid(ppID, member, bid);

525     Offer(ppID, member, bid);

526

527     **return true;**

528

529     // If the whole fund process succeed, the money bid must be converted to

530 ↪ percentage (bid/cost),

531     // so it will be possible to define the quota and the SEB a member has to gain.

532     // This is made on PowerUpResult(...).

533 }

534

535     // To update a member or a PP dataset on the ledger.

536 **private static object** Change( **object** id, **params object[]** opts )

537 {

538     // A referendum must start in case the change needs group's consensus.

539     **string** rID;

540

541     // If 'id' is a 'byte[]' == member.

542     **if** ( !IsValidId(id) )

543 {

544         **if** ( opts.Length != 0 )

545 {

546             // Only the member can change its own personal data.

547             // To UPDATE, the params must be [ 'profile option ', 'value ' ].

548             **if** ( !IsValidNum(opts[1]) )

549 {

550                 UpMemb((**byte[]**)id, (**string**)opts[0], (**string**)opts[1]);

551                 Update("Profile data.", id);

552                 **return true;**

553 }

## Code E.15 - GROUP FUNCTIONS (continuity).

```

554         // Any member can request the change of registration data of other member.
555         // To UPDATE, the params must be [ 'register option' , 'value' ].
556         rID = Ref( "Change register_ " , (string)opts[0] , (byte[])id , (int)opts[1] );
557         Process( rID , "Request the change of a member's registration data." );
558         return rID ;
559     }
560
561     // else
562     // Any member can request to delete another member.
563     rID = Ref( "Delete member_ " , null , (byte[])id );
564     Process(rID , "Request to dismiss a member." );
565     return rID ;
566 }
567
568     // Otherwise , the 'id' is a 'string' with prefix 'P' == power plant.
569
570     // Only the member can change its own bid.
571     // To UPDATE, the params must be [ 'address' , 'new bid value' ].
572     if ( opts.Length == 2 )
573     {
574         UpBid((string)id , (byte[]) opts[0] , (BigInteger)opts[1]);
575         Update("Bid." , id);
576         return true;
577     }
578
579     // Any member can request the change of the 'utility' a PP belongs to .
580     // To UPDATE, the params must be [ 'new utility name' ].
581     if ( opts.Length == 1 )
582     {
583         rID = Ref( "Change utility_ " , (string)opts[0] , ((string)id).AsByteArray() );
584         Process( rID , "Request the change of a PP's utility name." );
585         return rID ;
586     }
587
588     // Any member can request to DELETE a PP.
589     // The 'opts.Length' is empty.
590     rID = Ref("Delete PP_ " , null , ((string)id).AsByteArray());
591     Process(rID , "Request to delete a PP." );
592     return rID ;
593 }
594
595     // To integrate a new PP on the group power generation.
596     private static string PowerUp( int capacity , int cost , string utility , uint
597     ↪ timeToMarket )
598     {
599         string rID = Ref( Int2Str(capacity) , utility , "" . AsByteArray() , cost ,
600         ↪ timeToMarket );
601         Process( rID , "Request to add a new PP." );
602         return rID ;
603     }

```

## Code E.16 - GROUP FUNCTIONS (conclusion).

```

603  // To allow the transfer of shares/tokens from someone to someone else (
604  // → transactive energy indeed).
605  // The 'fromAddress' will exchange an amount of shares with 'toAddress' by a
606  // → defined token price,
607  // i.e., while 'fromAddress' sends shares to 'toAddress', the 'toAddress' sends
608  // → tokens to 'fromAddress'.
609  private static bool Trade( byte[] fromAddress, byte[] toAddress, BigInteger
610  // → exchange, BigInteger price )
611  {
612      int n = 2;
613      BigInteger[] toWallet = new BigInteger[n];
614      BigInteger[] fromWallet = new BigInteger[n];
615
616      for ( int r = 0; r < n; r++ )
617      {
618          // Remember: register = {"quota", "tokens"}.
619
620          fromWallet[r] = (BigInteger)GetMemb( fromAddress, register[r] );
621          toWallet[r] = (BigInteger)GetMemb( toAddress, register[r] );
622      }
623
624
625      if ( ( fromWallet[0] < exchange ) || ( toWallet[1] < price ) ) return false;
626
627      UpMemb(fromAddress, register[0], fromWallet[0] - exchange);
628      UpMemb(toAddress, register[0], toWallet[0] + exchange);
629
630      UpMemb(toAddress, register[1], toWallet[1] - price);
631      UpMemb(fromAddress, register[1], fromWallet[1] + price);
632
633      Transfer(fromAddress, toAddress, exchange, price);
634      return true;
635  }

```

## Code E.17 - ADMINISTRATIVE FUNCTIONS (to be continued).

```

639  {
640      // Calculates the result.
641      CalcResult(rID);
642
643      // Retrieves the address from private storage.
644      byte[] address = (byte[])GetRef(rID, "address");
645
646      if ( Str2Bool( (string)GetRef(rID, "outcome") ) )
647      {
648          // Retrieves the member data from private storage.
649          string fullName = (string)GetRef(rID, "proposal");
650          string utility = (string)GetRef(rID, "notes");
651
652          // Adds a new member after the group approval.
653          Member( address, fullName, utility, 0, 0 );
654          Membership( address, "Welcome on board!" );
655          return true;
656      }

```

## Code E.18 - ADMINISTRATIVE FUNCTIONS (continuity).

```

658     // Otherwise, leave the user out of the group.
659     Membership( address , "Not approved yet." );
660     DelMemb( address );
661     return false;
662 }
663
664 private static bool ChangeResult( string rID )
665 {
666     // Calculates the result.
667     CalcResult(rID);
668
669     if ( Str2Bool( (string)GetRef(rID , "outcome") ) )
670     {
671         Process(rID , "Approved.");
672
673         // Identifies the proposal and does the respective operation.
674         string proposal = (string)GetRef(rID , "proposal");
675
676         byte[] key;
677
678         if ( proposal == "Change register_")
679         {
680             key = (byte[])GetRef(rID , "address");
681             UpMemb(key , (string)GetRef(rID , "notes") , ( BigInteger )GetRef(rID , "cost"));
682             Update("Registration data." , key);
683         }
684
685         if ( proposal == "Delete member_")
686         {
687             key = (byte[])GetRef(rID , "address");
688             BigInteger portion = ( BigInteger )GetMemb(key , "quota");
689             BigInteger give_out = portion / (NumOfMemb() - 1);
690
691             foreach ( byte[] member in ListOfMembers() )
692             {
693                 // In an infinitesimal period of time the group will be disbalanced
694                 // until the related member be completely deleted.
695                 // There is no side effect on power distribution, and
696                 // it is better than iterate through each member.
697
698                 Distribute(member , give_out , 0);
699             }
700
701             DelMemb(key);
702             Membership(key , "Goodbye.");
703         }
704
705         if ( proposal == "Change utility_")
706         {
707             UpPP(rID , "utility" , (string)GetRef(rID , "notes"));
708             Update("Belonging of." , rID);
709         }

```

## Code E.19 - ADMINISTRATIVE FUNCTIONS (continuity).

```

711     if ( proposal == "Delete PP_" )
712     {
713         DelPP(rID);
714         Update("Deletion of.", rID);
715     }
716
717     return true;
718 }
719
720 Process(rID, "Denied.");
721 return false;
722 }
723
724 private static object PowerUpResult( string rID, string ppID = null )
725 {
726     // STEP 1 – Analyzes the referendum about the request for a new PP.
727     if ( ppID == null )
728     {
729         if ( isLock(rID) )
730             return Warning("There isn't a result about the new PP request yet.");
731
732         // After the 'timeFrameRef' waiting period...
733
734         // Evaluates the referendum result only once.
735         if ( (BigInteger)GetRef(rID) == 0 )
736         {
737             // Updates the result.
738             CalcResult(rID);
739
740             if ( Str2Bool( (string)GetRef(rID, "outcome") ) )
741             {
742                 // Referendum has succeeded. It's time to register a new PP.
743
744                 // Gets the terms from the begining of the process.
745                 string capacity = (string)GetRef(rID, "proposal");
746                 BigInteger cost = (BigInteger)GetRef(rID, "cost");
747                 string utility = (string)GetRef(rID, "notes");
748                 uint timeToMarket = (uint)GetRef(rID, "time");
749
750                 // Generates the PP ID.
751                 string PPid = PP(capacity, cost, utility, timeToMarket);
752
753                 // Starts to raise money for it.
754                 CrowdFunding(PPid);
755                 Process(PPid, "Shut up and give me money!");
756                 return PPid;
757             }
758
759             // Otherwise, the referendum of the PP request (Ref ID) continues
760             ← registered
761                 // in the group space, however it does not have a register (PP ID).
762                 Process(rID, "This PP was not approved yet. Let's wait a bit more.");
763                 return false;
764             }

```

## Code E.20 - ADMINISTRATIVE FUNCTIONS (continuity).

```

765         return "This process step is completed." ;
766     }
767
768     // STEP 2 – Analyzes the crowdfunding of the new PP approved .
769     if ( isLock(ppID) )
770         return Warning("There isn't a result about the new PP crowdfunding yet .");
771
772     // After the 'timeFrameCrowd' waiting period ...
773
774     // Keeps the value for the following operations handy .
775     BigInteger target = (BigInteger)GetPP(ppID, "cost");
776
777     // Evaluates the crowdfunding result only once .
778     if ( (BigInteger)GetCrowd(ppID) == 0 )
779     {
780         // Updates the result .
781         UpCrowd(ppID, "hasresult", 1);
782
783         // Gets the value from the crowdfunding process .
784         BigInteger funding = (BigInteger)GetCrowd(ppID, "totalamount");
785
786         // Evaluates if the building of the new PP starts or not .
787         if ( funding == target )
788         {
789             // Crowdfunding has succeeded .
790             UpCrowd(ppID, true);
791
792             // Updates the number of investors .
793             UpPP(ppID, "numOffundMemb", ListOfFunders(ppID).Length);
794
795             Process(ppID, "New power plant on the way .");
796             return true;
797         }
798
799         // Otherwise , the "success" remains as 'false' .
800         foreach ( byte[] funder in ListOfFunders(ppID) )
801         {
802             Cancel(ppID, funder);
803         }
804
805         Process(ppID, "Fundraising has failed .");
806         return false;
807     }
808
809     // STEP 3 – Analyzes the PP operation status .
810
811     // Calculates the date the new PP is planned to start to operate ,
812     // that can be always updated until the deadline .
813
814     // operationDate = ICO_endTime + PP_timeToMarket
815     uint operationDate = (uint)GetCrowd(ppID, "endtime") + (uint)GetPP(ppID, "
     ↳ timetomarket");

```

## Code E.21 - ADMINISTRATIVE FUNCTIONS (continuity).

```

817     if ( InvokedTime() <= operationDate )
818         return Warning("The new PP is not ready to operate yet.");
819
820         // After waiting for the time to market...
821
822         // Evaluates the construction only once.
823         if ( (BigInteger)GetPP(ppID) == 0 )
824         {
825             // When the PP is ready to operate, it's time to distribute tokens and shares
826             .
827
828             // Increases the total power supply of the group.
829             BigInteger capOfPP = (BigInteger)GetPP(ppID, "capacity");           // [MW]
830             BigInteger capOfGroup = TotalSupply() + capOfPP;                      // [MW]
831             Storage.Put("totalsupply", capOfGroup);
832
833             // Identifies how much the new PP takes part on the group total power supply.
834             BigInteger sharesOfPP = capOfPP/capOfGroup;                         // [pu]
835
836             foreach ( byte[] funder in ListOfFunders(ppID) )
837             {
838                 // Gets the member contribution.
839                 BigInteger grant = GetBid(ppID, funder);                          // [R$]
840
841                 // Identifies the member participation rate.
842                 BigInteger rate = grant/target; // [pu]
843
844                 // Defines how much of crypto-currency a member acquires from the new PP's
845                 // capacity.
846                 BigInteger tokens = (rate * capOfPP)/factor;                     // [MW/1000 = kW == SEB
847                 .
848
849                 // Defines how much of energy a member is entitled over the total power
850                 // supply.
851                 BigInteger quota = rate * sharesOfPP * capOfGroup;                // [MW]
852
853                 // Updates the member register data.
854                 Distribute(funder, quota, tokens);
855             }
856
857             // Updates the result.
858             UpPP(ppID, "hasstarted", 1);
859
860             Process(ppID, "A new power plant is now operating.");
861             return true;
862         }
863
864         // To return the IDs of each PP.
865         private static byte[][][] ListOfPPs()
866         {
867             byte[][][] ppIDs = new byte[ (int)NumOfPP() ][];

```

## Code E.22 - ADMINISTRATIVE FUNCTIONS (conclusion).

```

868     for ( int num = 0; num < NumOfPP(); num++ )
869     {
870         ppIDs[num] = PPData.ID.Get( Int2Str(num+1) );
871     }
872
873     return ppIDs;
874 }
875
876 // To return the address of each member.
877 private static byte[][] ListOfMembers()
878 {
879     byte[][] addresses = new byte[ (int)NumOfMemb() ][];
880
881     for ( int num = 0; num < NumOfMemb(); num++ )
882     {
883         addresses[num] = MemberData.ID.Get( Int2Str(num+1) );
884     }
885     return addresses;
886 }
887
888 // To return a list of members that have financed a given PP.
889 private static byte[][] ListOfFunders( string ppID )
890 {
891     byte[][] funders = new byte[ (int)GetCrowd(ppID, "contributions") ][];
892
893     BigInteger bid;
894     int num = 0;
895     foreach ( byte[] member in ListOfMembers() )
896     {
897         bid = GetBid( ppID, member );
898
899         if ( bid != 0 )
900         {
901             funders[num] = member;
902             num++;
903         }
904     }
905
906     return funders;
907 }
```

## Code E.23 - SYSTEM FUNCTIONS (to be continued).

```

918     {
919         BigInteger[] pastWallet = new BigInteger[ register.Length ];
920         int num = 0;
921
922         // Remember: register = {"quota", "tokens"}.
923         foreach ( string data in register )
924         {
925             pastWallet[num] = ( (BigInteger)GetMemb(toAddress, data) );
926             num++;
927         }
```

## Code E.24 - SYSTEM FUNCTIONS (continuity).

```

929     UpMemb(toAddress, register[0], pastWallet[0] + quota);
930     UpMemb(toAddress, register[1], pastWallet[1] + tokens);
931     Transfer(null, toAddress, quota, tokens);
932 }
933
934 // To create a custom ID of a process based on its particular specifications.
935 private static string ID( string prefix, bool unique, params string[] args )
936 {
937     // Assuming that all operations are little-endian.
938
939     // STEP 1 - Creates the hash.
940     string data = null;
941
942     if ( unique ) data = Int2Str((int)InvokedTime());
943
944     foreach ( string a in args )
945     {
946         data = String.Concat(data, a);
947     }
948
949     byte[] scriptHash = Hash160( data.AsByteArray() );           // length = 20 bytes
950
951     // STEP 2 - Enlarges the array to get the desired BigInteger's numbers range.
952     byte[] temp = scriptHash.Take(1);
953     scriptHash = scriptHash.Concat(temp);                         // length = 21 bytes
954
955     // STEP 3 - Adds the prefix.
956     byte[] preID = scriptHash.Concat( prefix.AsByteArray() );    // length = 22
957     ← bytes
958
959     // STEP 3 - Converts to Base58.
960     return Encode58( preID );
961 }
962
963 // To properly store a boolean variable.
964 private static string Bool2Str( bool val )
965 {
966     if ( val ) return "1";
967     return "0";
968 }
969
970 // To properly read a boolean from storage.
971 private static bool Str2Bool( string val )
972 {
973     if ( val == "1" ) return true;
974     return false;
975 }
976
977 // To affordably convert an integer to a string.
978 private static string Int2Str( int num, string s = null )
979 {
980     if ( num == 0 ) return s;
981
982     int quotient = num / 10;
983     int remainder = num % 10;
984
985     string trick = Digits()[ remainder ];

```

## Code E.25 - SYSTEM FUNCTIONS (continuity).

```

986     return Int2Str(quotient, String.Concat(trick, s));
987 }
988
989 // The Base58 encoding scheme.
990 private static string Encode58( byte[] preID )
991 {
992     // Restricts to positive values.
993     byte[] data = preID.Concat("\x00".AsByteArray());    // length = 23 bytes
994
995     // Converts 'byte[]' to 'BigInteger' and then to 'int'.
996     int input = (int) data.ToBigInteger();
997
998     // Defines the variables for the encode.
999     int[] result = new int[40]; // Big value to avoid constraints.
1000     int basis = 58;
1001     int pos = 0;
1002     int quotient = basis+1;
1003
1004     // Starts the encode with the Base58 indexes.
1005     while (quotient > basis)
1006     {
1007         quotient = input / basis;
1008         result[pos] = input % basis;
1009         input = quotient;
1010         pos++;
1011     }
1012     result[pos] = input;
1013
1014     // Converts the array of indexes to 'string'.
1015     string b58 = null;
1016     for (int k=pos; k >= 0; k--)
1017     {
1018         b58 += Alphabet[ result[k] ];
1019     }
1020
1021     return b58;
1022 }
1023
1024 // To evaluate if an object is a 'string' that may represent both a PP ID or a
1025 // Ref ID.
1026 private static bool IsValidId( object id )
1027 {
1028     return ( (((string)id)[0] == 'P') || (((string)id)[0] == 'R') );
1029 }
1030
1031 // To evaluate if an object can be converted to 'BigInteger'.
1032 private static bool IsValidNum( object test )
1033 {
1034     string temp = (string)test;
1035
1036     foreach (char t in temp)
1037     {
1038         foreach (string d in Digits())
1039         {
1040             if (t.ToString() == d) return false;
1041         }
1042     }

```

## Code E.26 - SYSTEM FUNCTIONS (continuity).

```

1042     return true;
1043 }
1044
1045 // To filter the relationships between members and PPs.
1046 private static void ShowContributedValues( object lookForID, object[] listOfIDs )
1047 {
1048     BigInteger bid;
1049
1050     // Displays all the members and their contributions to a given PP crowdfunding.
1051     if ( IsValidId(lookForID) )
1052     {
1053         // The 'lookForID' is a PP ID.
1054
1055         // Gets members' bid by a PP funding process.
1056         foreach ( byte[] member in listOfIDs )
1057         {
1058             bid = GetBid((string)lookForID, member);
1059
1060             if ( bid != 0 )
1061             {
1062                 Runtime.Notify( new object[] { member, bid } );
1063             }
1064         }
1065     }
1066
1067     // Displays all the PPs and its funds from a specific member.
1068     else
1069     {
1070         // The 'lookForID' is a member address.
1071
1072         // Gets PPs by a member investments.
1073         foreach ( string ppID in listOfIDs )
1074         {
1075             bid = GetBid(ppID, (byte[])lookForID);
1076
1077             if ( bid != 0 )
1078             {
1079                 Runtime.Notify( new object[] { ppID, bid } );
1080             }
1081         }
1082     }
1083 }
1084
1085 // To calculate the referendum result only once.
1086 private static void CalcResult( string rID )
1087 {
1088     if ( Str2Bool( (string)GetRef(rID) ) )
1089     {
1090         BigInteger totalOfVotes = (BigInteger)GetRef(rID, "numofvotes");
1091         BigInteger totalOfTrues = (BigInteger)GetRef(rID, "counttrue");
1092
1093         if ( totalOfTrues > (totalOfVotes / 2) )
1094         {
1095             // Referendum has succeeded.
1096             UpRef(rID, true);
1097         }
1098         // Otherwise, the "outcome" remains as 'false'.

```

## Code E.27 - SYSTEM FUNCTIONS (conclusion).

```

1100      // Certifies the calculation happens only once.
1101      UpRef(rID, "hasresult", 1);
1102  }
1103 }
1104
1105 // Actually, it restricts a given operation to happen based on a timestamp.
1106 // Before a given time frame, no one is allowed to continue the process.
1107 // The monitoring of the time happens off-chain.
1108 // Once the time stated is reached, any member can then resume the process.
1109 private static bool isLock( string id )
1110 {
1111     uint endTime;
1112
1113     if ( id[0] == 'R' )
1114     {
1115         endTime = (uint)GetRef(id, "endtime");
1116     }
1117     else
1118     {
1119         // id[0] == 'P'
1120         endTime = (uint)GetCrowd(id, "endtime");
1121     }
1122
1123     if ( InvokedTime() <= endTime ) return true;
1124     return false;
1125 }
```

## Code E.28 - Exception, a highlight of the System Functions.

```

1127 // The restrictive message to show up.
1128 private static bool Warning(string msg = "Only members can access this
1129    ↳ information. Join us!")
1130 {
1131     Exception(msg);
1132     return false;
1133 }
```

## Code E.29 - CRUD methods for MEMBERS (to be continued).

```

1137 // --> create
1138 private static void Member( byte[] address, string fullName, string utility,
1139    ↳ BigInteger quota, BigInteger tokens )
1140 {
1141     MemberData.FullName.Put(address, fullName);
1142     MemberData.Utility.Put(address, utility);
1143     MemberData.Quota.Put(address, quota);
1144     MemberData.Tokens.Put(address, tokens);
1145
1146     // Increases the total number of members.
1147     BigInteger temp = NumOfMemb() + 1;
1148     Storage.Put("numofmemb", temp);
1149
1150     // Stores the address of each member.
1151     MemberData.ID.Put( Int2Str((int)temp), address );
1152 }
```

Code E.30 - CRUD methods for MEMBERS (continuity).

```

1153 // --> read
1154 private static object GetMemb( byte[] address, string opt = "fullname" )
1155 {
1156     if ( opt == "utility" ) return MemberData.Utility.Get(address);
1157     else if ( opt == "quota" ) return MemberData.Quota.Get(address);
1158     else if ( opt == "tokens" ) return MemberData.Tokens.Get(address);
1159     else return MemberData.FullName.Get(address);
1160 }
1161
1162 // --> update
1163 // Detailed restrictions to update 'profile' or 'register' data are set
1164 // on the function 'Change'. Here this feature is handled by polymorphism.
1165 private static bool UpMemb( byte[] address, string opt, string val )
1166 {
1167     // Doesn't invoke Put if value is unchanged.
1168     string orig = (string)GetMemb(address, opt);
1169     if ( orig == val ) return true;
1170
1171     // Use Delete rather than Put if the new value is empty.
1172     if ( val.Length == 0 )
1173     {
1174         DelMemb(address, opt);
1175         return true;
1176     }
1177
1178     // else
1179     if ( opt == "fullname" ) MemberData.FullName.Put(address, val);
1180     if ( opt == "utility" ) MemberData.Utility.Put(address, val);
1181
1182     return true;
1183 }
1184
1185 private static bool UpMemb( byte[] address, string opt, BigInteger val )
1186 {
1187     // Doesn't invoke Put if value is unchanged.
1188     BigInteger orig = (BigInteger)GetMemb(address, opt);
1189     if ( orig == val ) return true;
1190
1191     // Use Delete rather than Put if the new value is zero.
1192     if ( val == 0 )
1193     {
1194         DelMemb(address, opt);
1195         return true;
1196     }
1197
1198     // else
1199     if ( opt == "quota" ) MemberData.Quota.Put(address, val);
1200     if ( opt == "tokens" ) MemberData.Tokens.Put(address, val);
1201
1202     return true;
1203 }

```

Code E.31 - CRUD methods for MEMBERS (conclusion).

```

1205 // --> delete
1206 private static void DelMemb( byte[] address, string opt = "" )
1207 {
1208     // To support an economic action for the update method.
1209     if ( opt == "fullname" ) MemberData.FullName.Delete(address);
1210     else if ( opt == "utility" ) MemberData.Utility.Delete(address);
1211     else if ( opt == "quota" ) MemberData.Quota.Delete(address);
1212     else if ( opt == "tokens" ) MemberData.Tokens.Delete(address);
1213
1214     // The member exits the group database (opt == "").
1215     else
1216     {
1217         foreach ( string option in new string[] { "fullname", "utility", "quota", "
1218             tokens" } )
1219         {
1220             DelMemb(address, option);
1221         }
1222
1223         // Looks for the member 'key' (that may vary during the life cycle of the
1224             group).
1225         for ( int num = 1; num < NumOfMemb() + 1; num++ )
1226         {
1227             var index = Int2Str(num);
1228
1229             if ( address == MemberData.ID.Get(index) )
1230             {
1231                 // Wipes off the address of the member.
1232                 MemberData.ID.Delete(index);
1233
1234                 // Updates the following indexes.
1235                 while ( num <= NumOfMemb() )
1236                 {
1237                     num++;
1238                     var newIndexSameAddress = MemberData.ID.Get( Int2Str(num) );
1239                     MemberData.ID.Put( Int2Str(num - 1), newIndexSameAddress );
1240                 }
1241
1242                 // Ends the for loop.
1243                 break;
1244             }
1245         }
1246
1247         // Decreases the total number of members.
1248         BigInteger temp = NumOfMemb() - 1;
1249         Storage.Put("numofmemb", temp);
1250     }
1251 }
```

Code E.32 - CRUD methods for POWER PLANTS (to be continued).

```

1254 // --> create
1255 private static string PP( string capacity, BigInteger cost, string utility, uint
1256   ↵ timeToMarket )
1257 {
1258     // Creates the unique identifier.
1259     string id = ID( "\x53", true, new string[] {capacity, Int2Str((int)cost),
1260       ↵ utility, Int2Str((int)timeToMarket)} );
1261
1262     // Stores the practical values.
1263     PPData.Capacity.Put(id, capacity);
1264     PPData.Cost.Put(id, cost);
1265     PPData.Utility.Put(id, utility);
1266     PPData.TimeToMarket.Put(id, timeToMarket);
1267
1268     // Just states the other values since it is expensive to store null values.
1269     // PPData.NumOfFundMemb.Put(id, 0);
1270     // PPData.HasStarted.Put(id, 0);
1271
1272     // Increases the total number of PP units.
1273     BigInteger temp = NumOfPP() + 1;
1274     Storage.Put("numofpp", temp);
1275
1276     // Stores the ID of each PP.
1277     PPData.ID.Put( Int2Str((int)temp), id );
1278
1279     // Notifies about the creation of the PP ID.
1280     Process(id, "New PP registered.");
1281     return id;
1282 }
1283
1284 // --> read
1285 private static object GetPP( string ppID, string opt = "hasstarted" )
1286 {
1287     if ( opt == "capacity" ) return PPData.Capacity.Get(ppID);
1288     else if ( opt == "cost" ) return PPData.Cost.Get(ppID);
1289     else if ( opt == "utility" ) return PPData.Utility.Get(ppID);
1290     else if ( opt == "timetomarket" ) return PPData.TimeToMarket.Get(ppID);
1291     else if ( opt == "numoffundmemb" ) return PPData.NumOfFundMemb.Get(ppID);
1292     else return PPData.HasStarted.Get(ppID);
1293 }
1294
1295 // --> update
1296 // The 'utility', the 'hasstarted', and the 'timetomarket' are the only options
1297 // that can be changed.
1298 // However, the 'utility' can be changed anytime, the 'hasstarted' can be changed
1299 // only once, while
1300 // the 'timetomarket' is restricted by its deadline of start operation date.
1301 // To update the other options, delete the current PP and create a new one.
1302 private static void UpPP( string ppID, string opt, object val )
1303 {
1304     if ( opt == "utility" )

```

Code E.33 - CRUD methods for POWER PLANTS (continuity).

```

1301    {
1302        // Doesn't invoke Put if value is unchanged.
1303        string orig = (string)GetPP(ppID, "utility");
1304        if ( orig == (string)val ) return;
1305
1306        // Does nothing if the new value is empty.
1307        if ( ((string)val).Length == 0 ) return;
1308
1309        // else
1310        PPData.Utility.Put(ppID, (string)val);
1311        // WARNING: Logic constraints!
1312        // When the PP utility name changes, it should update each member utility
1313        //→ name as well.
1314        // However, only the member her/himself can change this information.
1315        // Therefore, 'utility' of both member's and PP's dataset must pointer to a
1316        //→ common database.
1317        // THIS WAS NOT IMPLEMENTED!
1318    }
1319
1320    if ( opt == "hasstarted" )
1321    {
1322        // Doesn't invoke Put if value is unchanged.
1323        BigInteger orig = (BigInteger)GetPP(ppID, "hasstarted");
1324        if ( orig == (BigInteger)val ) return;
1325
1326        // Does nothing if the new value is empty.
1327        if ( (BigInteger)val == 0 ) return;
1328
1329        // else
1330        PPData.HasStarted.Put(ppID, (BigInteger)val);
1331    }
1332
1333    if ( opt == "timetomarket" )
1334    {
1335        // Doesn't invoke Put if value is unchanged.
1336        uint orig = (uint)GetPP(ppID, "timetomarket");
1337        if ( orig == (uint)val ) return;
1338
1339        // Does nothing if the new value is empty.
1340        if ( (uint)val == 0 ) return;
1341
1342        // Does nothing if the deadline has passed by.
1343        uint deadline = (uint)GetCrowd(ppID, "endtime") + (uint)GetPP(ppID, "
1344        //→ timetomarket");
1345
1346        if ( InvokedTime() > deadline )
1347        {
1348            Warning("The time has passed by. You can no longer postpone it.");
1349            return;
1350        }

```

Code E.34 - CRUD methods for POWER PLANTS (conclusion).

```

1349      // else
1350      PPData.TimeToMarket.Put(ppID, (uint)val);
1351    }
1352  }
1353
1354  // --> delete
1355  private static void DelPP( string ppID )
1356  {
1357    PPData.Capacity.Delete(ppID);
1358    PPData.Cost.Delete(ppID);
1359    PPData.Utility.Delete(ppID);
1360    PPData.TimeToMarket.Delete(ppID);
1361    if ( (BigInteger)GetPP(ppID, "hasstarted") != 0 ) PPData.HasStarted.Delete(ppID
1362    ↵ );
1362    if ( (BigInteger)GetPP(ppID, "numoffundmemb") != 0 ) PPData.NumOfFundMemb.
1363    ↵ Delete(ppID);
1363
1364    // Looks for the PP 'key' (that may vary during the life cycle of the group).
1365    for ( int num = 1; num < NumOfPP() + 1; num++ )
1366    {
1367      var index = Int2Str(num);
1368
1369      if ( ppID == PPData.ID.Get(index).AsString() )
1370      {
1371        // Wipes off the ID of the PP.
1372        PPData.ID.Delete(index);
1373
1374        // Updates the following indexes.
1375        while (num <= NumOfMemb())
1376        {
1377          num++;
1378          var newIndexSameId = PPData.ID.Get( Int2Str(num) );
1379          PPData.ID.Put( Int2Str(num - 1), newIndexSameId );
1380        }
1381
1382        // Ends the for loop.
1383        break;
1384      }
1385    }
1386
1387    // Decreases the total power supply of power plants.
1388    BigInteger temp = TotalSupply() - (BigInteger)GetPP(ppID, "capacity");
1389    Storage.Put("totalsupply", temp);
1390
1391    // Decreases the total number of power plant units.
1392    temp = NumOfPP() - 1;
1393    Storage.Put("numofpp", temp);
1394  }

```

Code E.35 - CRUD methods for REFERENDUMS (to be continued).

```

1399 // --> create
1400 private static string Ref( string proposal, string notes, byte[] address, int
1401   ⇔ cost = 0, uint time = 0 )
1402 {
1403     string id = ID( "\x5A", true, new string[] {proposal, notes, Int2Str(cost)} );
1404
1405     // Stores the practical values.
1406     RefData.Proposal.Put(id, proposal);
1407     RefData.Notes.Put(id, notes);
1408     RefData.Outcome.Put(id, Bool2Str(false));
1409     RefData.StartTime.Put(id, InvokedTime());
1410     RefData.EndTime.Put(id, InvokedTime() + timeFrameRef);
1411
1412     // Evaluates the values before stores them since it is expensive to store null
1413     ⇔ values.
1414     if ( address.Length != 0 ) RefData.Address.Put(id, address);
1415     if ( cost != 0 ) RefData.Cost.Put(id, cost);
1416     if ( time != 0 ) RefData.Time.Put(id, time);
1417
1418     // Just states the other values since it is expensive to store null values.
1419     // RefData.MoneyRaised.Put(id, 0);
1420     // RefData.NumOfVotes.Put(id, 0);
1421     // RefData.CountTrue.Put(id, 0);
1422     // RefData.HasResult.Put(id, 0);
1423
1424     // Increases the total number of referendum processes.
1425     BigInteger temp = NumOfRef() + 1;
1426     Storage.Put("numoref", temp);
1427
1428     // Stores the ID of each referendum.
1429     RefData.ID.Put( Int2Str((int)temp), id );
1430
1431     // Notifies about the creation of the referendum ID.
1432     Process(id, "The referendum process has started.");
1433     return id;
1434 }
1435
1436 // The function to vote on a referendum is declared above.
1437
1438 // --> read
1439 private static object GetRef( string rID, string opt = "hasresult" )
1440 {
1441     if ( opt == "proposal" ) return RefData.Proposal.Get(rID);
1442     else if ( opt == "notes" ) return RefData.Notes.Get(rID);
1443     else if ( opt == "cost" ) return RefData.Cost.Get(rID);
1444     else if ( opt == "address" ) return RefData.Address.Get(rID);
1445     else if ( opt == "time" ) return RefData.Time.Get(rID);
1446     else if ( opt == "moneyraised" ) return RefData.MoneyRaised.Get(rID);
1447     else if ( opt == "numofvotes" ) return RefData.NumOfVotes.Get(rID);
1448     else if ( opt == "counttrue" ) return RefData.CountTrue.Get(rID);
1449     else if ( opt == "outcome" ) return RefData.Outcome.Get(rID);
1450     else if ( opt == "starttime" ) return RefData.StartTime.Get(rID);
1451     else if ( opt == "endtime" ) return RefData.EndTime.Get(rID);
1452     else return RefData.HasResult.Get(rID);
1453 }
```

Code E.36 - CRUD methods for REFERENDUMS (conclusion).

```

1453 // --> update
1454 // It is only possible to internally change the 'moneyraised', the 'numofvotes',
1455 // the 'counttrue', the 'hasresult' and the 'outcome' (polymorphism).
1456 private static void UpRef( string rID, string opt, BigInteger val )
1457 {
1458     if ( (opt == "numofvotes") || (opt == "moneyraised") || (opt == "counttrue") ||
1459     → (opt == "hasresult") )
1460     {
1461         BigInteger orig = (BigInteger)GetRef(rID, opt);
1462
1463         // Doesn't invoke Put if value is unchanged.
1464         if ( orig == val ) return;
1465
1466         if ( val == 0 )
1467         {
1468             // Deletes the respective storage if the new value is zero.
1469             if ( opt == "numofvotes" ) RefData.NumOfVotes.Delete(rID);
1470             else if ( opt == "moneyraised" ) RefData.MoneyRaised.Delete(rID);
1471             else if ( opt == "counttrue" ) RefData.CountTrue.Delete(rID);
1472             else RefData.HasResult.Delete(rID);
1473         }
1474         else
1475         {
1476             // Update the respective storage with the new value.
1477             if ( opt == "numofvotes" ) RefData.NumOfVotes.Put(rID, val);
1478             else if ( opt == "moneyraised" ) RefData.MoneyRaised.Put(rID, val);
1479             else if ( opt == "counttrue" ) RefData.CountTrue.Put(rID, val);
1480             else RefData.HasResult.Put(rID, val);
1481         }
1482     }
1483
1484 private static void UpRef( string rID, bool val )
1485 {
1486     bool orig = Str2Bool( (string)GetRef(rID, "outcome") );
1487
1488     // Doesn't invoke Put if value is unchanged.
1489     if ( orig == val ) return;
1490
1491     // else
1492     RefData.Outcome.Put(rID, Bool2Str(val));
1493 }
1494
1495 // --> delete
1496 // A referendum process remains forever... and ever.

```

Code E.37 - CRUD methods for CROWDFUNDING (to be continued).

```

1501 // --> create
1502 private static void CrowdFunding( string ppID )
1503 {
1504     // Stores the practical values.
1505     ICOData.StartTime.Put(ppID, InvokedTime());
1506     ICOData.EndTime.Put(ppID, InvokedTime() + timeFrameCrowd);
1507     ICOData.Success.Put(ppID, Bool2Str(false));
1508
1509     // Just states the other values since it is expensive to store null values.
1510     // ICOData.TotalAmount.Put(ppID, 0);
1511     // ICOData.Contributions.Put(ppID, 0);
1512     // ICOData.HasResult.Put(ppID, 0);
1513 }
1514
1515 // The function to bid on a crowdfunding is declared above.
1516 // However, the option 'ICOData.Bid.Put(bidID, value)' is only available through
1517 // the updating method, and not as part of the creation method.
1518
1519 // --> read
1520 private static BigInteger GetBid( string ppID, byte[] member )
1521 {
1522     string bidID = ID( "\x27", false, new string[] {ppID, member.AsString()} );
1523     return ICOData.Bid.Get(bidID).AsBigInteger();
1524 }
1525
1526 private static object GetCrowd( string ppID, string opt = "hasresult" )
1527 {
1528     if ( opt == "starttime" ) return ICOData.StartTime.Get(ppID);
1529     else if ( opt == "endtime" ) return ICOData.EndTime.Get(ppID);
1530     else if ( opt == "totalamount" ) return ICOData.TotalAmount.Get(ppID);
1531     else if ( opt == "contributions" ) return ICOData.Contributions.Get(ppID);
1532     else if ( opt == "success" ) return ICOData.Success.Get(ppID);
1533     else return ICOData.HasResult.Get(ppID);
1534 }
1535
1536 // --> update
1537 private static void UpBid( string ppID, byte[] member, BigInteger bid )
1538 {
1539     BigInteger orig = GetBid(ppID, member);
1540
1541     // Doesn't invoke Put if bid is unchanged OR the new value is zero.
1542     if ( (orig == bid) || (bid == 0) ) return;
1543
1544     // else
1545     string bidID = ID( "\x27", false, new string[] {ppID, member.AsString()} );
1546     ICOData.Bid.Put( bidID, orig + bid );
1547 }
1548
1549 // Only the 'totalamount', 'contributions', 'hasresult' and 'success' (
1550 // → polymorphism) can be updated.
1551 private static void UpCrowd( string ppID, string opt, BigInteger val )
1552 {
1553     if ( (opt == "totalamount") || (opt == "contributions") || (opt == "hasresult")
1554         → )
1555     {
1556         BigInteger orig = (BigInteger)GetCrowd(ppID, opt);

```

## Code E.38 - CRUD methods for CROWDFUNDING (conclusion).

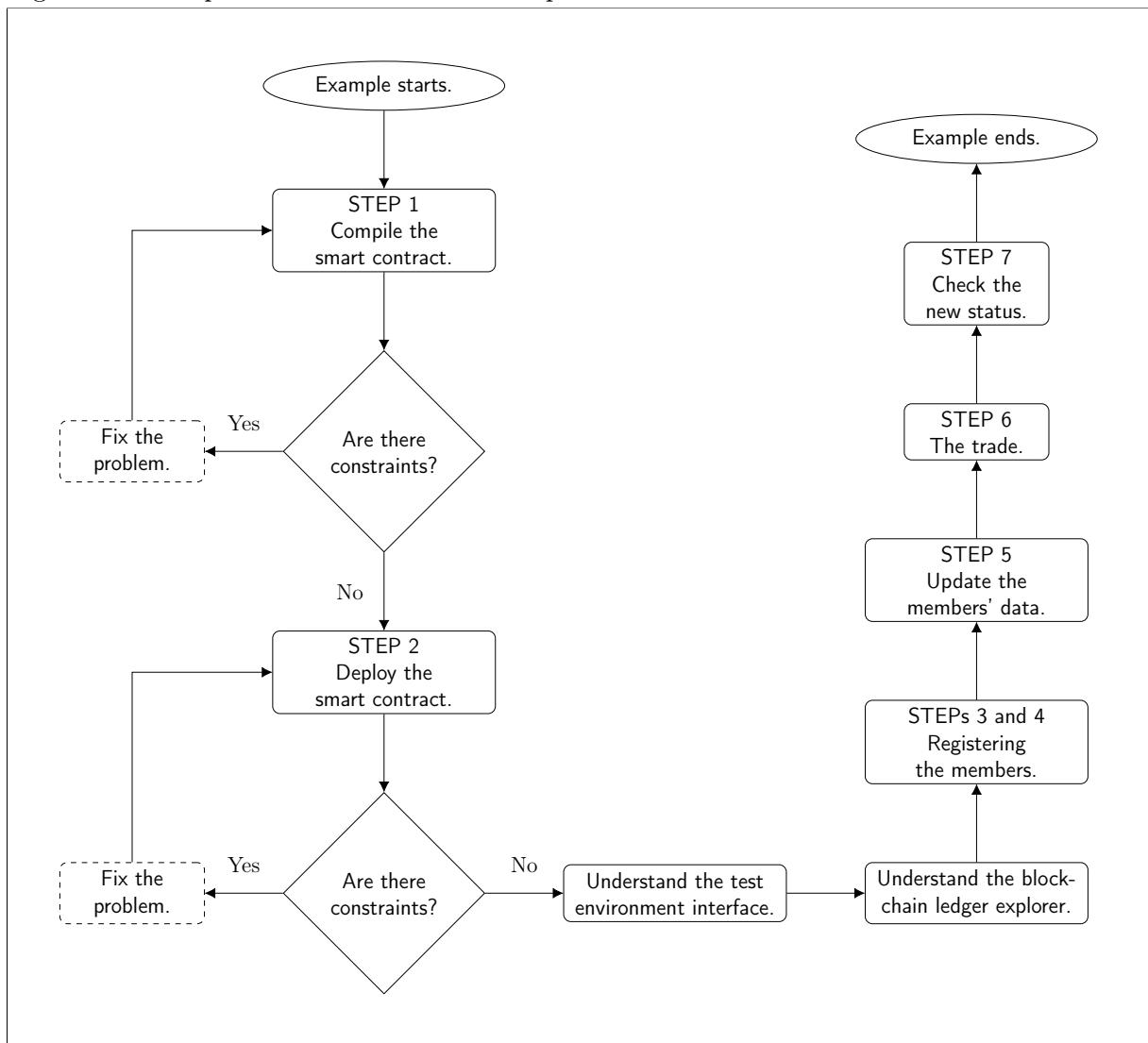
```

1556     // Doesn't invoke Put if value is unchanged.
1557     if ( orig == val ) return;
1558
1559     // Deletes the respective storage if the new value is zero.
1560     if ( val == 0 )
1561     {
1562         if ( opt == "totalamount" ) ICOData.TotalAmount.Delete(ppID);
1563         else if ( opt == "contributions" ) ICOData.Contributions.Delete(ppID);
1564         else ICOData.HasResult.Delete(ppID);
1565     }
1566     else
1567     {
1568         // Updates the respective storage with the new value.
1569         if ( opt == "totalamount" ) ICOData.TotalAmount.Put(ppID, val);
1570         else if ( opt == "contributions" ) ICOData.Contributions.Put(ppID, val);
1571         else ICOData.HasResult.Put(ppID, val);
1572     }
1573 }
1574 }
1575
1576 private static void UpCrowd( string ppID, bool val )
1577 {
1578     string orig = (string)GetCrowd(ppID, "success");
1579
1580     // Doesn't invoke Put if value is unchanged.
1581     if ( orig == Bool2Str(val) ) return;
1582
1583     // else
1584     ICOData.Success.Put(ppID, Bool2Str(val));
1585 }
1586
1587 // --> delete
1588 private static void Cancel( string ppID, byte[] member )
1589 {
1590     BigInteger grant = GetBid(ppID, member);
1591
1592     // Decreases the total amount of funds.
1593     BigInteger funds = (BigInteger)GetCrowd(ppID, "totalamount");
1594     UpCrowd(ppID, "totalamount", funds - grant);
1595
1596     // Decreases the total number of contributions.
1597     BigInteger contributions = (BigInteger)GetCrowd(ppID, "contributions");
1598     UpCrowd(ppID, "contributions", contributions - 1);
1599
1600     // Deletes the member's offer.
1601     string bidID = ID( "\x27", false, new string[] { ppID, member.ToString() } );
1602     ICOData.Bid.Delete(bidID);
1603
1604     // Notifies about the cancel of the bid.
1605     Retract( ppID, member, 0, (-1 * grant) );
1606 }
1607
1608 // A crowdfunding process remains forever... even if it fails.
1609 // In this case, only the 'totalamount' and 'contributions' will
1610 // be "deleted" through the function above.
1611 }
```

## APPENDIX F – The setup of the TE example

The following steps precede the example described in Section 3.4. The objective here is showcase the full process involved until reach the simple transaction of power exchange. Moreover, all the considerations addressed on chapters about stakeholders relationships, blockchain platform specifications, energy market structure, and software requirements have a practical importance now. As can be seen in the upcoming lines, some variables had to be adjusted and some workarounds had to be made to reach the desired outcome. However, while some modifications had no big impact in the example purpose, others may considerably affect the application performance. The presentation of these adaptations closes the appendix.

Figure 24 - The process to conduct the example.



Source: The author, 2019.

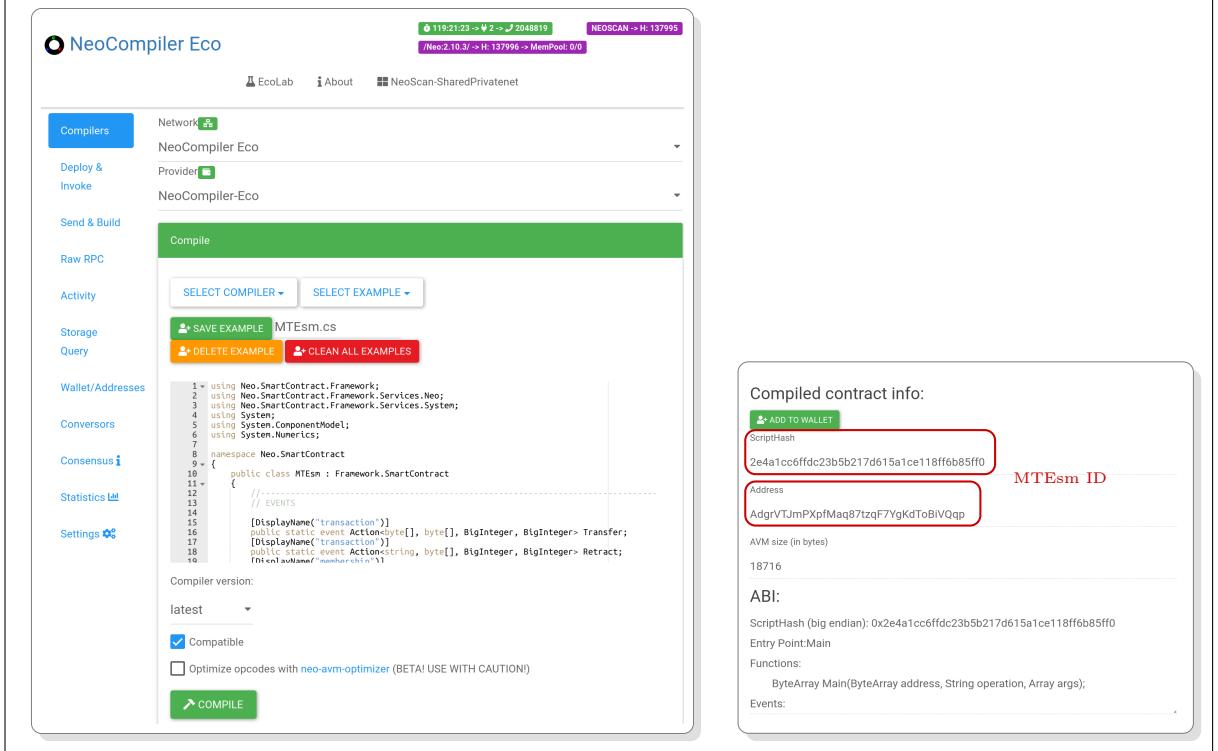
Additionally, Figure 24 summarizes the example course detailed in the following

steps. Starting with the MTEsm compilation and identification of its record in the ledger, the example moves to the steps of registering the users in the group's DApp in order to properly identify the members for future exchanges. Next, changes on members' registers are made to reflect the same information shown in Table 1b (Chapter 1) since the long power plant registering process is avoided. Although members need tokens to define the price exchange, and the creation of SEBs is dependent on the power plant capacity, for simplicity, an alternative course was used to give a symbolic reward for a member to conclude the trade process without being labeled as a donation. The closing step is the TE indeed, with further analysis besides the outcome described in the example presented in Section 3.4.

## STEP 1 The MTEsm compilation

The use of the DApp capability starts with the success of the MTEsm compilation at the Neo VM in order to get a valid public address for the smart contract, as shown in Figure 25. To run the example in a reasonable time, the variable *timeFrameRef* was changed to 120 seconds (Code E.5, l.103, Appendix E).

Figure 25 - Essential steps before MTEsm deployment.



The figure consists of two side-by-side screenshots of the NeoCompiler Eco web application.

**(a) Smart contract code placed in the compiler.** This screenshot shows the 'Compile' section of the NeoCompiler Eco interface. A code editor displays the following C# code for the MTEsm smart contract:

```

1  using Neo.SmartContract.Framework;
2  using Neo.SmartContract.Framework.Services.Neo;
3  using Neo.SmartContract.Framework.Services.System;
4  using System;
5  using System.ComponentModel;
6  using System.Numerics;
7
8  namespace Neo.SmartContract
9  {
10     public class MTEsm : Framework.SmartContract
11     {
12         // EVENTS
13
14         [DisplayName("transaction")]
15         public static event Action<byte[], byte[], BigInteger, BigInteger> Transfer;
16
17         [DisplayName("retract")]
18         public static event Action<string, byte[], BigInteger, BigInteger> Retract;
19         [EventName("membershort")]
20

```

Below the code editor are several buttons: 'SELECT COMPILER' (dropdown), 'SELECT EXAMPLE' (dropdown), 'SAVE EXAMPLE' (button), 'DELETE EXAMPLE' (button), and 'CLEAN ALL EXAMPLES' (button). At the bottom are dropdown menus for 'Compiler version:' (set to 'latest') and checkboxes for 'Compatible' (checked) and 'Optimize opcodes with neo-avm-optimizer (BETA! USE WITH CAUTION)' (unchecked). A large green 'COMPILE' button is at the bottom right.

**(b) Part of the compilation output.** This screenshot shows the 'Compiled contract info:' section of the interface. It includes the following details:

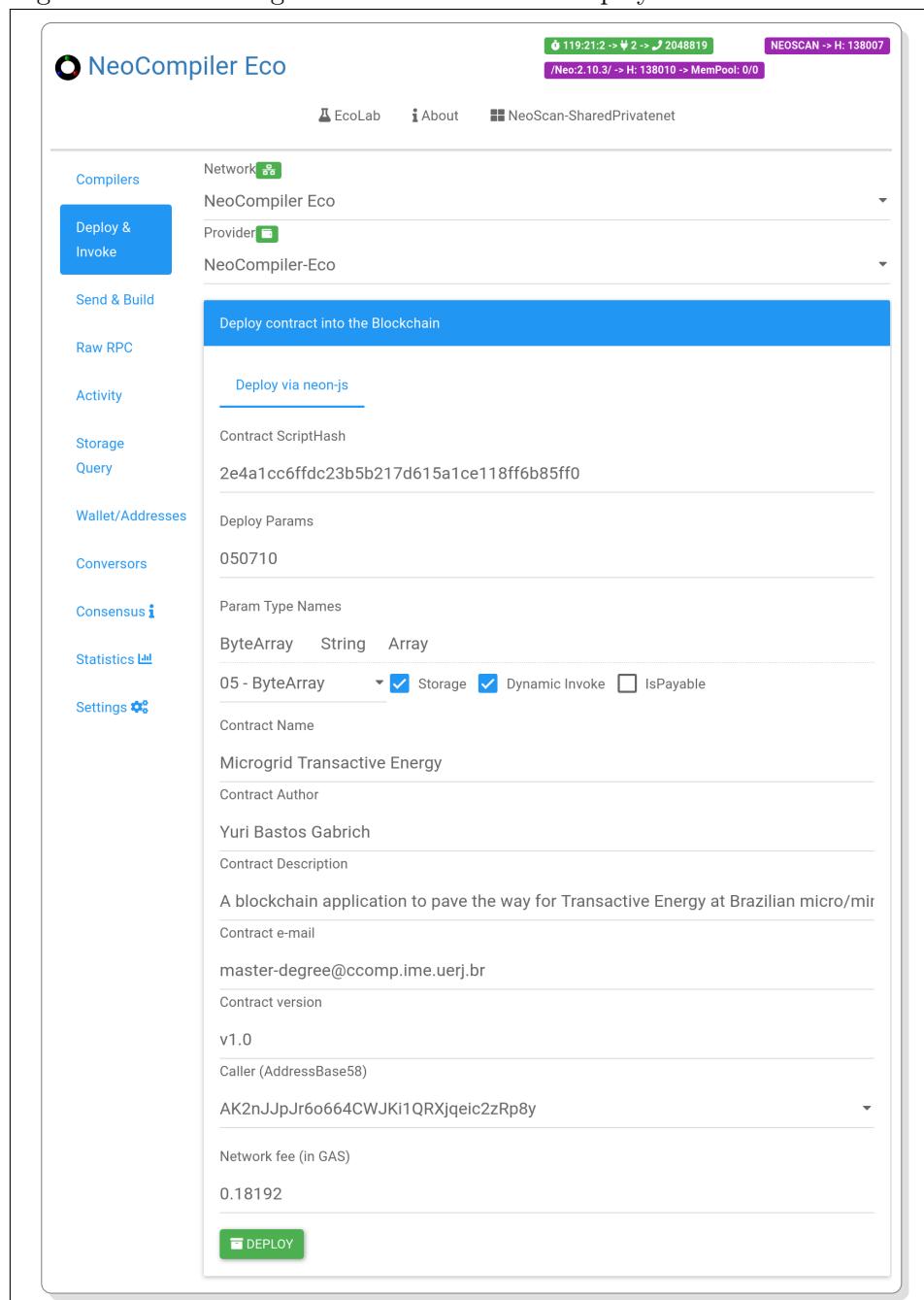
- MTEsm ID:** 2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0
- Address:** AdgrVTJmPxpfMaq87tzqF7YgKdToBiVQqp
- AVM size (in bytes):** 18716
- ABI:** ScriptHash (big endian): 0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0
- Entry Point:** Main
- Functions:** ByteArray Main(ByteArray address, String operation, Array args);
- Events:**

Source: The author, 2019.

## STEP 2 The MTEsm deploy

The next step is the deploy of the MTEsm in the blockchain network. Figure 26 states some configurations to customize the request and highlights the *Storage* check box, a required option to use the persistent space. The *Network fee* is immediately calculated and only charged if the transaction size is above 1024 bytes, even though it can be changed to incentive faster consensus outcome (CELIA; QIAN, 2019).

Figure 26 - The configuration of the MTEsm deployment.



Source: The author, 2019.

## **STEP 2.1** The activity Menu and JSONs

The information of the deploy request is concatenated in the *Activity* menu (Figure 27a), where is presented the transaction input parameters and its hash ID in the ledger. The latter links to a JSON file (reproduced in Figure 27b) with values of the input and output parameters that are used to create the new block state in the ledger. Moreover, the *Activity* interfaces with the Neo VM output API through the button that showcases the transaction *Status*, i.e., *HALT* for the Neo VM positive answer and *FAULT* for a negative one (just remembering that both states are recorded in the ledger and have system costs). When clicked, the *Status* button changes the window to the *Raw RPC* menu, where the transaction ID is already passed as an input for the VM log API (*getapplicationlog*), and displays another JSON file, but now in the same window and below the input parameters (Figure 28). This file contains, in addition to the essential identification of the transaction, two more outputs: (i) the *System fee* of 1000 GAS indicated by the key *gas\_consumed*, which is the cost to store the smart contract in the blockchain platform, and (ii) the return arguments of the operation, that have nothing to display for the deploy case. Upcoming representations of the RPC output will follow Figure 27b format instead of Figure 28 because of paper space limitation.

## **STEP 2.2** The NeoScan: finding the transactions in the ledger

To finish the deploy explanation, let's query the transaction in the blockchain ledger. All the images cited before have a button in the top labeled *NeoScan-SharedPrivateNet*, that opens the explorer interface of the EcoLab's ledger. The block height number (137,805) is available in Figure 27a, and the search by it returns a list of transaction IDs included in this new state of the blockchain network (Figure 29). The two types of transactions that compose the related block refer to the information about (i) the deploy, named as *Invocation* (Figure 30a); and (ii) the system fee payment, named as *Miner* (Figure 30b). Figure 30a has the same information shown in Figure 27b, regardless of the scripts, its details that are hidden in the image, and the difference in the value of *gas\_consumed* (probably due to equivocal setup of the test environment faced with different versions of the Neo blockchain). The advantage of this view is the clear payment course of the deploy operation by the user `AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y`, i.e., the *System Fee* payment of 990 GAS is the difference between the values before (*Sent from*) and after (*Sent to*) the operation, plus the *Network Fee*. Figure 30b complements the financial route exhibiting the address of who has received the *Network Fee* payment of 0.18192 GAS. In summary, this overview can be reproduced in every next step, and although only this one is shown, it is worth to look into them by yourself.

Figure 27 - The identification of the outputs of the deploy.

ID	Status	Restore	Transaction parameters	Height	Hash
0	HALT		{ "contract_scripthash": "2e4a1cc6ffd...d615a1ce118ff6b85ff0", "caller": "AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y", "mynetfee": 0.18192, "contract_script": "56c56b6c766b00527ac46c766b...7cc4766a51c3517cc4616c7566", "storage": 3, "returntype": "05", "par": "050710", "contract_description": "A blockchain application to pave the way for Transactive Energy at Brazilian micro/mini-grids", "contract_email": "master-degree@ccomp.ime.uerj.br", "contract_author": "Yuri Bastos Gabrich", "contract_version": "v1.0", "contract_appname": "Microgrid Transactive Energy", "type": "deploy" }	137805	4571b1...1db60a

(a) Response.

```
{
  "vouts": [
    {
      "value": 534093.582316,
      "txid": "dd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d",
      "n": 0,
      "asset": "GAS",
      "address_hash": "AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y"
    }
  ],
  "vin": [
    {
      "value": 535083.764236,
      "txid": "088864374e29f31dc9982e91c0b0dfa1fd95c31c8a30f1dff805590fed7c3e0",
      "n": 0,
      "asset": "GAS",
      "address_hash": "AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y"
    }
  ],
  "version": 1,
  "type": "InvocationTransaction",
  "txid": "dd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d",
  "time": 1591968292,
  "sys_fee": 990.0,
  "size": 19141,
  "scripts": [
    {
      "verification": "21031a6c6fbddf02ca351745fa86b9ba5a9452d785ac4f7fc2b7548ca2a46c4fcf4aac",
      "invocation": "403031a6b0c38f487ea0cfa87e5d2017035448b5e42192d7e13104616f68b7b7f9642d1f48755d2
        ↳ e6e65123af97b4b83088805b9668b22a161538166ee00e1e18d"
    }
  ],
  "pubkey": null,
  "nonce": null,
  "net_fee": 0.18192,
  "description": null,
  "contract": null,
  "claims": [
  ],
  "block_height": 137805,
  "block_hash": "4571b1aeb91496ab15111885a9ecbc3967b8cf47385c96795115a5209c1db60a",
  "attributes": [
  ],
  "asset": null
}
```

(b) Output (JSON).

Legend: The highlight in (a) is the number to search the deploy of the MTEsm in the ledger.

Source: The author, 2019.

Figure 28 - The RPC output of the deploy.

The screenshot shows the NeoCompiler Eco web interface. The left sidebar has a blue header "Raw RPC" and lists various options like Compilers, Deploy & Invoke, Send & Build, Activity, Storage Query, Wallet/Addresses, Conversors, Consensus, Statistics, and Settings. The main area has tabs for Network (selected), Provider, NeoCompiler, and NeoCompiler-L. A blue bar at the top says "Invoke neo nodes (JSON RPC)". Below it, a dropdown menu says "Select a method". The "RPC Json Model" section contains a JSON object with a red box around the first line: [{"jsonrpc": "2.0", "id": 5, "method": "getapplicationlog", "params": "dd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d"}]. The "Output RPC" section shows the JSON response with several red boxes highlighting specific fields: "txid" (with value "0xdd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d"), "gas\_consumed" (with value "1000"), and "stack" (with value "[]"). At the bottom are two buttons: a green one with a green icon and a purple one with a purple icon labeled "NOTIFICATIONS".

Legend: The first highlight is the API method for the transaction ID `dd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d` (hidden in the box) to get the deploy output. The other highlights are the values available in the ledger (Figure 30a).

Source: The author, 2019.

Figure 29 - The deploy transaction block.

**Block Information**

Block | 4571b1aeb91496ab15111885a9ecbc3967b8cf47385c96795115a5209c1db60a

<a href="#">Back to all blocks</a>		<a href="#">Previous block</a>	<a href="#">Next block</a>
Block index <b>137,805</b>	Transactions <b>2</b>	Hash <b>4571b1aeb91496ab15111885a9ecbc3967b8cf47385c96795115a5209c1db60a</b>	
Time <b>11 minutes ago</b>	Version <b>0</b>	Merkle root <b>de29fe85dbf0c557228c88d8148a78c8b8c1d09da2a3f5643d462e02cee1aac2</b>	
Validator <b>AZ81H31DMWzbSnFDLFkzh9vHwaDLayV7fU</b>	Size <b>19,653 bytes</b>	Generated GAS <b>8</b>	

## Transactions

Type	Transaction ID	Completed on
Miner	<a href="#">f0ec55f8c087254d8d1bba3a694f123825f3dd7f5410c5694c7628ea709ace11...</a>	12-06-2020   10:24:52:000
Invocation	<a href="#">dd5a764b3a429acc6dd5d6fe32a348198348bc1b458aa28742e1c216db95ec8d...</a>	12-06-2020   10:24:52:000

Transactions 0 to 2 of 2

Legend: This block has the two transactions indicated in Figure 30a and Figure 30b. Note the original image colours were replaced to better adjust to the printing format.

Source: The author, 2019.

Figure 30 - The transactions details of the MTEsm deploy in the ledger.


dd5a764b3a429acct
Back to all transactions

## Transaction Information

**Invocation | Hash: dd5a764b3a429acct**

Sent from	Sent to
<a href="#">AK2nJpJr6o664CWJKi1QRXjqsic2zRp8y</a> 535,083.764236 GAS	<a href="#">AK2nJpJr6o664CWJKi1QRXjqsic2zRp8y</a> 534,093.582316 GAS
Time <b>18 minutes ago</b>	Network Fee <b>0.18192</b>
Included in Block <a href="#">137,805</a>	System Fee <b>990</b>
Size <b>19,141 bytes</b>	

(a) The invocation payment.


Block Height, Hash,..
Back to all transactions

## Transaction Information

**Miner | Hash: f0ec55f8c087254d8d1bba3a694f123825f3dd7f5410c5694c7628ea709ace11**

Sent to
<a href="#">AWLYWWXB8C9Lt1nHdDZJnC5cpYJgRDLk17</a> 0.18192 GAS
Time <b>20 minutes ago</b>
Network Fee <b>0</b>
Included in Block <a href="#">137,805</a>
System Fee <b>0</b>
Size <b>70 bytes</b>

(b) The miner payment.

Legend: The highlights indicate the same values available through the API (Figure 28).

Source: The author, 2019.

### STEP 3 The first admission process

This is the first relevant operation that can be performed at this moment of the MTEsm lifetime since any other operation will return empty or default values, and this represents waste of money. So, the member  $M1$  was chosen to fulfill this task. Figure 31 and Figure 32 display the input and output for the admission of  $M1$  that has no referendum request at this initial process.

Figure 31 - Admission request of  $M1$  (to be continued).

The screenshot shows the 'Interacting with a Smart Contract - Invoke' interface. At the top, it says 'Invoke via neon-js'. Below that, 'Contract ScriptHash' is set to '2e4a1cc6ffdc23b5b217d615'. The 'System fee (GAS)' is '0' and the 'Network fee (GAS)' is also '0'. Under 'Attach NEO' and 'Attach GAS', both are '0'. In the 'Function to Invoke' section, 'use param array' is checked. The 'Main' function is selected. The parameters are: Param 1:  in array (Type: Addr, Value: AK2nJJpJr6o664), Param 2:  in array (Type: String, Value: admission), Param 3:  in array (Type: String, Value: M1). Param 4:  in array (Type: String, Value: Traditional Energy), Param 5:  in array (Type: None, Value: Script Param 5), Param 6:  in array (Type: None, Value: Script Param 6). Param 7:  in array (Type: None, Value: Script Param 7). Below this, 'Caller (AddressBase58)' is set to 'AK2nJJpJr6o664CWJKi1Q'. There is a 'UPLOAD ABI' button and a 'Choose File' button with 'No file chosen'. At the bottom, there are three buttons: '(INVOKE)', '+ ADD PARAMS', and 'Advancing signing' with a checkbox.

Legend: The invoke operation for the first member. The highlighted box is a handy interface available by the test environment to write the JSON parameters.

Source: The author, 2019.

For simplicity, Figure 31 was clipped to only show the bottom of the page of the *Deploy & Invoke* menu with the invoke option, that is available right below the deploy option as shown in Figure 26. The highlighted area in the picture is a helpful interface to write the *Invoke Parameters (JSON)*. It defines the calling function (*Main*), and the way its arguments must be written, otherwise the VM returns *FAULT*. For the MTEsm, the main function requires for the first argument an user address formatted as byte array, for the second one, a string indicating which operation to perform, and for the third, an array with arguments for the related operation (Code E.6, l.134, Appendix E). Although

useful, this detailed view is avoided in the next figures since they can be interpreted in the *Invoke Parameters (JSON)* box.

Figure 32 - Admission request of *M1* (conclusion).

```
[{"jsonrpc": "2.0",
 "id": 5,
 "result": {
 "txid": "0x87762463cee49140f7c6593788e002ded48f41932e1ac3f614dc2b03b049ba",
 "executions": [
 {
 "trigger": "Application",
 "contract": "0xa7c893245fb673b65e51e8f01e62cabd315adf58",
 "vmstate": "HALT",
 "gas_consumed": "8.212",
 "stack": [
 {
 "type": "Integer",
 "value": "1"
 }
 ],
 "notifications": [
 {
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "6d656d62657273686970"
 },
 {
 "type": "ByteArray",
 "value": "23ba2703c53263e8d6e522dc32203339dc8ee9"
 },
 {
 "type": "ByteArray",
 "value": "57656c636f6d65206f6e20626f61726421"
 }
 ]
 }
 ]
 }
 ]}]}
```

Legend: Output (JSON RPC).

Source: The author, 2019.

## STEP 4 The second admission process

Differently than the step before, this admission process takes longer, and the new request to be part of the group (Figure 33) need to be approved by more than 50% of the members. In this case, the *M1* vote (Figure 34) is decisive to approve *M3* (Figure 35). On the other hand, this step has a similar financial cost than the one before, i.e., it is not required to transfer *NEO coins* but a transaction fee in *GAS* exist and must be payed by whoever request the invoke transaction.

### STEP 4.1 An analysis of the referendum

The referendum process is the core management of a shareable consumption group, and the present case has the democratic principle of one vote, one value. Here, the example starts really easy with the final decision up to only one vote, as shown before in Figure 34 and Figure 35. However, the next step has an entirely different scenario, since both members have to agree to get the desired outcome or, otherwise, they get stuck in time. The greater the number of members, the greater the challenge for agreement, even with a transparent process to do so.

Figure 33 - Admission request of M3.

The figure consists of two parts, (a) and (b), illustrating the admission request process using the Neon.js interface.

(a) Invoke:

This section shows the user interface for interacting with a smart contract via neon.js. The interface includes fields for Contract ScriptHash (2e4a1cc6ffdc23b5b217d615), System fee (GAS) (0), Network fee (GAS) (0), and Invoke Parameters (JSON) [{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"admission"}, {"type":16,"value": [{"type":7,"value":"M3"}, {"type":7,"value":"Traditional Energy System"}]}]. There are also fields for Caller (AddressBase58) (APLJBPhRg2XLhtpxEHd6:•) and ABI upload (Choose File). Buttons for INVOKE, ADD PARAMS, and Advancing signing are present.

(b) Output (JSON RPC):

```
[{"jsonrpc": "2.0", "id": 5, "result": { "txid": "0x4ce58d7bdf024d7595a46ebab3c4fbff63544c9ac9a392fd72298a8b593313e9", "executions": [ { "trigger": "Application", "contract": "0x484f819767d4dcfa29f12b1fd6917814eac64bc9", "vmstate": "HALT", "gas_consumed": "3.533", "stack": [ { "type": "ByteArray", "value": "525a45594d436f61594c416b50714370617750676f524a6b523567413369" } ], "notifications": [ { "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0", "state": { "type": "Array", "value": [ { "type": "ByteArray", "value": "70726f63657373" }, { "type": "ByteArray", "value": "525a45594d436f61594c416b50714370617750676f524a6b523567413369" }, { "type": "ByteArray", "value": "546865207265666572656e64756d2070726f636573732068617320737461727465642e" } ] } ], "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0", "state": { "type": "Array", "value": [ { "type": "ByteArray", "value": "6d656d62657273686970" }, { "type": "ByteArray", "value": "52eaab8b2aab608902c651912db34de36e7a2b0f" }, { "type": "ByteArray", "value": "5265717565737420666f722061646d697373696f6e2e" } ] } } ] } } ] }
```

(b) Output (JSON RPC):

Source: The author, 2019.

Figure 34 - Vote of  $M1$  in the referendum about the admission request of  $M3$ .

(a) Invoke.

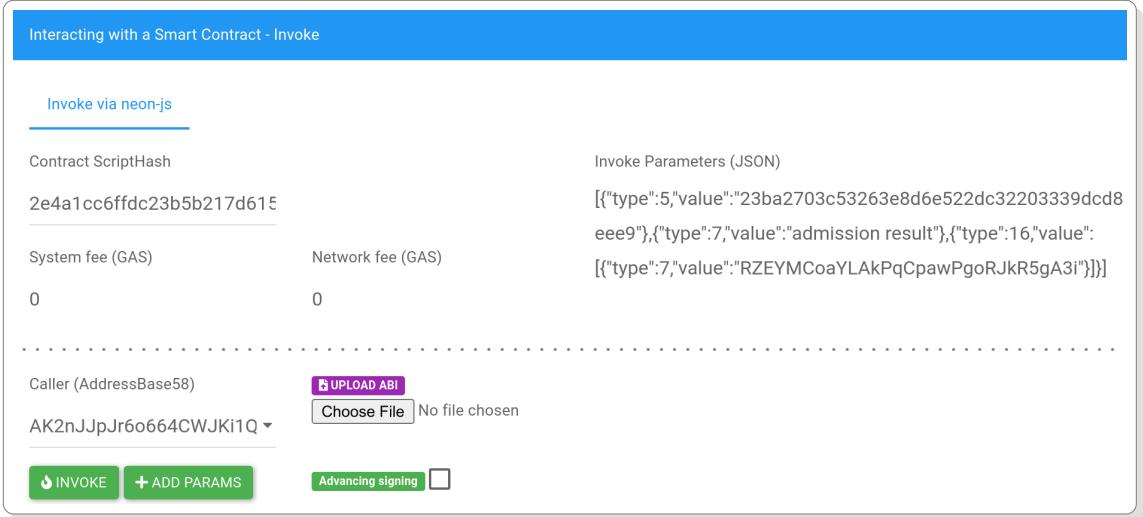
```
[{"jsonrpc": "2.0",
"id": 5,
"result": {
"txid": "0x95bce7374ff3b58545ef192a9db9e9e6bb03e3166cc9a987f0265eb2741bc249",
"executions": [
{
"trigger": "Application",
"contract": "0xd1b629240445f501dc8b4ee943ed452086ed23f",
"vmstate": "HALT",
"gas_consumed": "4.033",
"stack": [
{
"type": "Integer",
"value": "1"
}
],
"notifications": [
{
"contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
"state": {
"type": "Array",
"value": [
{
"type": "ByteArray",
"value": "62616c6c6f74"
},
{
"type": "ByteArray",
"value": "525a45594d436f61594c416b50714370617750676f524a6b523567413369"
},
{
"type": "ByteArray",
"value": "23ba2703c53263e8d6e522dc32203339cd8eee9"
},
{
"type": "ByteArray",
"value": "74727565"
}
]
}
]
}
]
}
}]}}
```

(b) Output (JSON RPC).

Source: The author, 2019.

Figure 35 - Admission result for the *M3* request.

**(a) Invoke.**



The screenshot shows the "Interacting with a Smart Contract - Invoke" interface. It has two main sections: "Contract ScriptHash" (2e4a1cc6ffdc23b5b217d615) and "Invoke Parameters (JSON)". The JSON parameters are:

```
[{"type":5,"value":"23ba2703c53263e8d6e522dc32203339cd8eee9"}, {"type":7,"value":"admission result"}, {"type":16,"value":[]}, {"type":7,"value":"RZEYMCoaYLakPqCpawPgoRJkR5gA3!"}]
```

Below these are fields for "System fee (GAS)" (0), "Network fee (GAS)" (0), "Caller (AddressBase58)" (AK2nJJpJr6o664CWJKi1Q), and a file upload section ("Choose File" button, "No file chosen"). At the bottom are buttons for "INVOKE" and "ADD PARAMS".

**(b) Output (JSON RPC).**



The screenshot shows the JSON RPC output. The response is:

```
[{"jsonrpc": "2.0", "id": 5, "result": { "txid": "0xe8f5109de54db8ea5fb02a47cddefe968516fffce9e884b33992192c896063bf", "executions": [ { "trigger": "Application", "contract": "0xdda3a80d300a09171fc345b96331625e3e953984", "vmstate": "HALT", "gas_consumed": "7.549", "stack": [ { "type": "Integer", "value": "1" } ], "notifications": [ { "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0", "state": { "type": "Array", "value": [ { "type": "ByteArray", "value": "6d656d62657273686970" }, { "type": "ByteArray", "value": "52eaab8b2aab608902c651912db34de36e7a2b0f" }, { "type": "ByteArray", "value": "57656c636f6d65206f6e20626f61726421" } ] } ] } ] }}
```

Source: The author, 2019.

## STEP 4.2 Understanding the return arguments

Another aspect to take a look at is the different types of outputs. All the RPC reproduced throughout the example steps are consequence of only positive VM outputs, which has two kinds of return arguments. The return value itself is identified by the key *stack* in the JSON. It is always a tuple with the returning type and value of the operation invoked. And the *notification* kind is identified by the same name key in the JSON but its tuple carries for each of its arguments both the contract ID and the *state* generated by it.

Moreover, the *stack* output represents a function return argument, i.e., a void function will return a null tuple, while a Boolean function with a false returning value will show a tuple with a pair of arguments with one of them being null. While the *notification* represents any other kind of printing assigned to run-time operations. Indeed, this is commonly used for the *event* method, which is a way to communicate an application that something has happened in the blockchain (CELIA, 2019a), like the notification in Figure 35b about the membership approved, that has its behaviour defined in Code E.2 (l.19, Appendix E). Also, the MTEsm use notifications to handle with exceptions through a function to complement it, as shown in Code E.28 (Appendix E). Nonetheless, the values are always presented in hexadecimal format in the JSON. The *Conversors* menu aids with the decode of *Strings* into UTF-8 text and numbers into *BigIntegers*.

Furthermore, the return arguments are not encrypted, neither appended on the ledger. They are stored on a side database with access through an open API, which means that anyone can monitor the transactions outputs based on the return arguments of the operations. For instance, while *M3* remains as a general user, she/he cannot process the *admission result* operation. At this time, the unique way she/he can check the referendum process outcome is through the *summary* operation. However, if the member *M1* invoke any of the operations, either a restricted or not, *M3* can get the operation output through the API request *getapplicationlog* if she/he knows the transaction ID belonging to the operation. This task is easy to perform since the identification is available in the ledger, as previously stated in Figure 29. But differently than the invoke transaction, the API has no cost, which means that *M3* can get the return output for free even not being allowed to perform the operation. Therefore, the DApp development must be sharpened.

## STEP 5 The update of the members' register data

Now, the change of the members' register is the last step before the TE operation. Nonetheless, the effort is longer than three times the step before, since it is needed one more vote process for each of the three updatings. The following sub-steps depict the quota change of the two members and the updating of  $M3$ 's token. Also, it presents the independent ability of members to request the change of other member's registering values, as shown in STEP 5.3.

### STEP 5.1 Updating of the $M1$ 's quota

Member  $M1$  has a share of 40% and the updating of MTEsm records to reflect this value is shown from Figure 36 to Figure 40.

Figure 36 - The request to change  $M1$ 's quota (to be continued).

The screenshot shows a web-based interface for interacting with a smart contract via neon-js. The top bar is blue with the title 'Interacting with a Smart Contract - Invoke'. Below it, there's a section titled 'Invoke via neon-js'.

**Contract ScriptHash:** 2e4a1cc6ffdc23b5b217d615

**System fee (GAS):** 0

**Network fee (GAS):** 0

**Invoke Parameters (JSON):**

```
[{"type":5,"value":"23ba2703c53263e8d6e522dc32203339cd8eee9"}, {"type":7,"value":"change"}, {"type":16,"value": [{"type":5,"value":"23ba2703c53263e8d6e522dc32203339cd8eee9"}, {"type":16,"value": [{"type":7,"value":"quota"}, {"type":2,"value":"40"}]}]}]
```

**Caller (AddressBase58):** AK2nJJpJr6o664CWJKi1Q ▾

**File Upload:** UPLOAD ABI Choose File No file chosen

**Action Buttons:** 🔥 INVOKE + ADD PARAMS Advancing signing [checkbox]

Legend: Invoke.

Source: The author, 2019.

Figure 37 - The request to change *M1*'s quota (conclusion).

```
{
  "jsonrpc": "2.0",
  "id": 5,
  "result": {
    "txid": "0x864b77f89875f49ad40a811e920ef04fc0016c000a43814035815634c8d016de",
    "executions": [
      {
        "trigger": "Application",
        "contract": "0xb44dd92d39572f615a19be855691c76c1f599168",
        "vmstate": "HALT",
        "gas_consumed": "4.621",
        "stack": [
          {
            "type": "ByteArray",
            "value": "52664c644c6769414c705268417a4a736e436a4c776d4757734276523779"
          }
        ],
        "notifications": [
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "70726f63657373"
                },
                {
                  "type": "ByteArray",
                  "value": "52664c644c6769414c705268417a4a736e436a4c776d4757734276523779"
                },
                {
                  "type": "ByteArray",
                  "value": "546865207265666572656e64756d2070726f636573732068617320737461727465642e"
                }
              ]
            }
          },
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "70726f63657373"
                },
                {
                  "type": "ByteArray",
                  "value": "52664c644c6769414c705268417a4a736e436a4c776d4757734276523779"
                },
                {
                  "type": "ByteArray",
                  "value": "5265717565737420746865206368616e6765206f662061206d656d62657227732072656769
→ 7374726174696f6e20646174612e"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

Legend: Output (JSON RPC).

Source: The author, 2019.

Figure 38 - *M1* vote in the referendum about the change request of *M1*.

(a) Invoke.

```
[{"jsonrpc": "2.0",
"id": 5,
"result": {
"txid": "0x68d47fd80ef2c55d1e13c46a23e76e4cd948f76520a27248e5f8ef7377c19ad5",
"executions": [
{
"trigger": "Application",
"contract": "0x0a1f041ff7901cd5d0d4adc0f6cf3984833face2",
"vmstate": "HALT",
"gas_consumed": "4.033",
"stack": [
{
"type": "Integer",
"value": "1"
}
],
"notifications": [
{
"contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
"state": {
"type": "Array",
"value": [
{
"type": "ByteArray",
"value": "62616c6c6f74"
},
{
"type": "ByteArray",
"value": "52664c644c6769414c705268417a4a736e436a4c776d4757734276523779"
},
{
"type": "ByteArray",
"value": "23ba2703c53263e8d6e522dc32203339dc8eee9"
},
{
"type": "ByteArray",
"value": "74727565"
}
]
}
]
}
]
}
}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

Figure 39 -  $M_3$  vote in the referendum about the change request of  $M_1$ .

(a) Invoke.

```
[{"jsonrpc": "2.0",
"id": 5,
"result": {
"txid": "0x8f01c32d155744b7c30475b88ff0d6892d0275c2f283c89548a08a1bdb61f17d",
"executions": [
{
"trigger": "Application",
"contract": "0xe62c662dcecf290409ec605d3f94425d597c78ce",
"vmstate": "HALT",
"gas_consumed": "4.033",
"stack": [
{
"type": "Integer",
"value": "1"
}
],
"notifications": [
{
"contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
"state": {
"type": "Array",
"value": [
{
"type": "ByteArray",
"value": "62616c6c6f74"
},
{
"type": "ByteArray",
"value": "52664c644c6769414c705268417a4a736e436a4c776d4757734276523779"
},
{
"type": "ByteArray",
"value": "52eaab8b2aab608902c651912db34de36e7a2b0f"
},
{
"type": "ByteArray",
"value": "74727565"
}
]
}
]
}
]
}
}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

Figure 40 - The result for the M1 change request.

**Interacting with a Smart Contract - Invoke**

**Invoke via neon-js**

Contract ScriptHash  
2e4a1cc6ffdc23b5b217d615

System fee (GAS)  
0

Network fee (GAS)  
0

Caller (AddressBase58)  
APLJBPhRg2XLhtpxEHd6...

**UPLOAD ABI**  
 Choose File No file chosen

**INVOKE** **+ ADD PARAMS** **Advancing signing**

Invoke Parameters (JSON)

```
[{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"change result"}, {"type":16,"value":[]}, {"type":7,"value":"RfLdLgiALpRhAzJsnCjLwmGWsBvR7y"}]]
```

(a) Invoke.

```
[{"jsonrpc": "2.0",
 "id": 5,
 "result": {
 "txid": "0x21d904d5279d67842432aa8b9dee3d0aa665d50bfd17f2e286ea04dea28dd6e4",
 "executions": [
 {
 "trigger": "Application",
 "contract": "0xd625562b68c707a4eaafc827237233aaa430994e",
 "vmstate": "HALT",
 "gas_consumed": "6.518",
 "stack": [
 {
 "type": "Integer",
 "value": "1"
 }
 ],
 "notifications": [
 {
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "70726f63657373"
 },
 {
 "type": "ByteArray",
 "value": "52664c644c6769414c705268417a4a736e436a4c776d4757734276523779"
 },
 {
 "type": "ByteArray",
 "value": "417070726f7665642e"
 }
 ]
 },
 {
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "6368616e6765"
 },
 {
 "type": "ByteArray",
 "value": "526567697374726174696f6e20646174612e"
 },
 {
 "type": "ByteArray",
 "value": "23ba2703c53263e8d6e522dc32203339dc8eee9"
 }
 ]
 }
 ]
 }
 ]}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

## STEP 5.2 Updating of the M3's quota

Member *M3* has a share of 60% and the updating of MTEsm records to reflect this value is shown from Figure 41 to Figure 45. Unexpectedly, Figure 44 catches a bug of the smart contract but it does not disturb the example's final outcome. On the other hand, it showcases the good behaviour of an exception.

Figure 41 - The request to change *M3*'s quota (to be continued).

The screenshot shows a web-based interface for interacting with a smart contract via neon-js. The top bar is blue with the title 'Interacting with a Smart Contract - Invoke'. Below this, there is a section titled 'Invoke via neon-js'.

**Contract ScriptHash:** 2e4a1cc6ffdc23b5b217d615

**System fee (GAS):** 0

**Network fee (GAS):** 0

**Invoke Parameters (JSON):**

```
[{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"change"}, {"type":16,"value": [{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":16,"value": [{"type":7,"value":{"quotas":60}}]}]]
```

**Caller (AddressBase58):** APLJBPhRg2XLhtpxEHd6...

**ABI:** UPLOAD ABI (button) Choose File (button) No file chosen

**Action Buttons:** 🔮 INVOKE (green button), + ADD PARAMS (green button), Advancing signing (checkbox)

Legend: Invoke.

Source: The author, 2019.

Figure 42 - The request to change *M3*'s quota (conclusion).

```
{
  "jsonrpc": "2.0",
  "id": 5,
  "result": {
    "txid": "0x4dfcc96f92f95cdd07c66b3d004563476057fe74b63ea0cb45e501a789270715",
    "executions": [
      {
        "trigger": "Application",
        "contract": "0xfbff0dd9072ec40c9d3d55c775a2f7afb931212c7",
        "vmstate": "HALT",
        "gas_consumed": "4.621",
        "stack": [
          {
            "type": "ByteArray",
            "value": "5253784e77336974526b343731774a3969705a4d705a594379564b505545"
          }
        ],
        "notifications": [
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "70726f63657373"
                },
                {
                  "type": "ByteArray",
                  "value": "5253784e77336974526b343731774a3969705a4d705a594379564b505545"
                },
                {
                  "type": "ByteArray",
                  "value": "546865207265666572656e64756d2070726f636573732068617320737461727465642e"
                }
              ]
            }
          },
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "70726f63657373"
                },
                {
                  "type": "ByteArray",
                  "value": "5253784e77336974526b343731774a3969705a4d705a594379564b505545"
                },
                {
                  "type": "ByteArray",
                  "value": "5265717565737420746865206368616e6765206f662061206d656d62657227732072656769
→ 7374726174696f6e20646174612e"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

Legend: Output (JSON RPC).

Source: The author, 2019.

Figure 43 - *M3* vote in the referendum about the change request of *M3*.

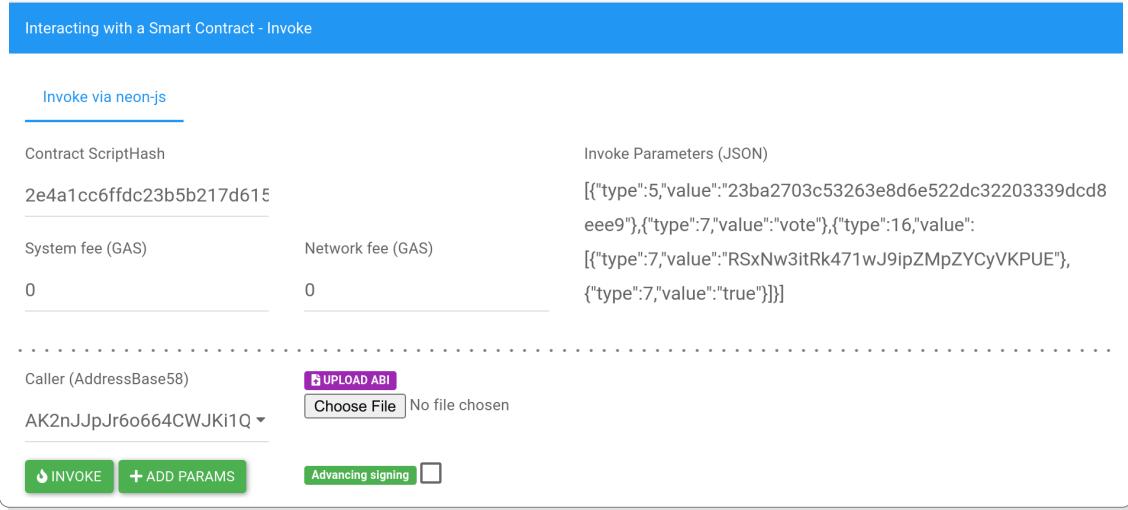
(a) Invoke.

```
[{"jsonrpc": "2.0",
 "id": 5,
 "result": {
 "txid": "0x6ad677e996016e0da6eadf235cc244037cb40eee38a23ccb46f2beb9c71566a",
 "executions": [
 {
 "trigger": "Application",
 "contract": "0xab7c455e7a2a7c75dc9e4e815bfb121545fa6522",
 "vmstate": "HALT",
 "gas_consumed": "4.033",
 "stack": [
 {
 "type": "Integer",
 "value": "1"
 }
 ],
 "notifications": [
 {
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "62616c6c6f74"
 },
 {
 "type": "ByteArray",
 "value": "5253784e77336974526b343731774a3969705a4d705a594379564b505545"
 },
 {
 "type": "ByteArray",
 "value": "52eaab8b2aab608902c651912db34de36e7a2b0f"
 },
 {
 "type": "ByteArray",
 "value": "74727565"
 }
 ]
 }
 ]
 }
 ]}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

Figure 44 -  $M1$  vote in the referendum about the change request of  $M3$ .



(a) Invoke.

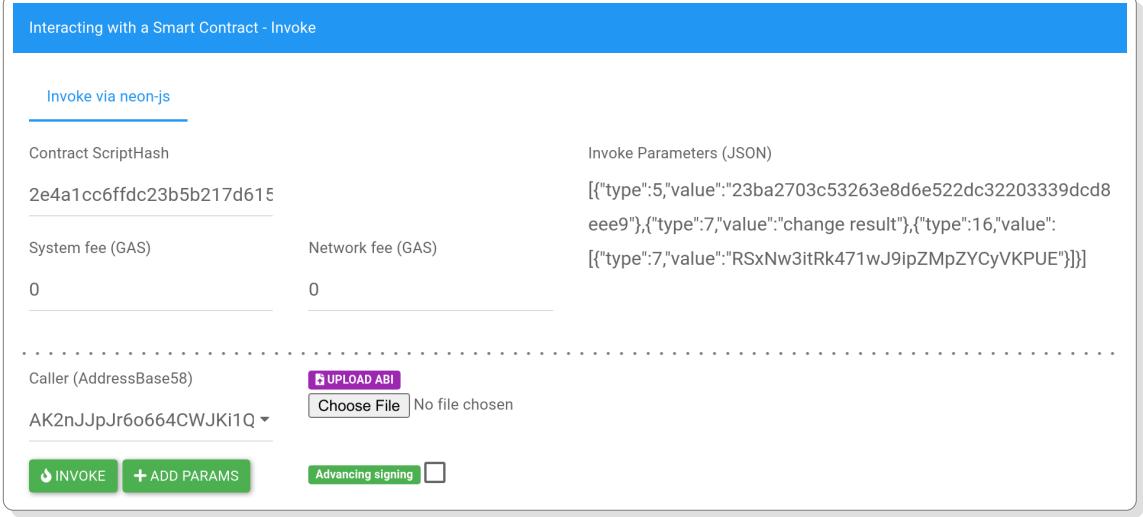
```
[{"jsonrpc": "2.0",
  "id": 5,
  "result": {
    "txid": "0x72f28ed68f785460caac5caa3dfa18534ec39720d9d6aab2503395fc66c639a5",
    "executions": [
      {
        "trigger": "Application",
        "contract": "0xf89010347b20cb90b8b979c04ad9e3a683c32751",
        "vmstate": "HALT",
        "gas_consumed": "0.86", ← paid operation
        "stack": [
          {
            "type": "ByteArray", ← "false"
            "value": ""
          }
        ],
        "notifications": [
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "696e76616c6964206f7065726174696f6e" ← "invalid operation"
                },
                {
                  "type": "ByteArray",
                  "value": "5468652062616c6f742068617320656e6465642e" ← "The ballot has ended."
                }
              ]
            }
          }
        ]
      }
    ]
  }
}]
```

(b) Output (JSON RPC).

Legend: The  $M1$  vote was invoked after the due date and the MTEsm exception returned the notifications highlighted. Also, note that the operation still had a cost and returned *false* to the ballot request, indicating the vote didn't succeed. Despite this,  $M3$ 's quota was updated with only 1 positive vote (50%), which indicates a failure in the code of the referendum process evaluation.

Source: The author, 2019.

Figure 45 - The result for the *M3* change request.



(a) Invoke.

```
[{"jsonrpc": "2.0",
 "id": 5,
 "result": {
 "txid": "0xa0b82d76146dfc83c9b7344ddec2407586eb34991767fcf8814d7b63f5d227fb",
 "executions": [
 {
 "trigger": "Application",
 "contract": "0x8f4cd86dead5278792b8dd672ce454e8ab978c52",
 "vmstate": "HALT",
 "gas_consumed": "6.518",
 "stack": [
 {
 "type": "Integer",
 "value": "1"
 }
 ],
 "notifications": [
 {
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "70726f63657373"
 },
 {
 "type": "ByteArray",
 "value": "5253784e77336974526b343731774a3969705a4d705a594379564b505545"
 },
 {
 "type": "ByteArray",
 "value": "417070726f7665642e"
 ]
 }
 ],
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "6368616e6765"
 },
 {
 "type": "ByteArray",
 "value": "526567697374726174696f6e20646174612e"
 },
 {
 "type": "ByteArray",
 "value": "52eaab8b2aab608902c651912db34de36e7a2b0f"
 ]
 }
 }
 ]}]]}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

### STEP 5.3 Updating of the *M1*'s tokens

Member *M1* was chosen to have the fictitious value of 5 SEBs because she/he has energy debit in the example described in Chapter 1. So, it will be advantageous for her/him to acquire part of *M3* shares. The updating of MTEsm records to allow this is shown from Figure 46 to Figure 50.

Figure 46 - The request of *M3* to change *M1*'s tokens (to be continued).

Interacting with a Smart Contract - Invoke

Invoke via neon.js

Contract ScriptHash  
2e4a1cc6ffdc23b5b217d615

System fee (GAS)  
0

Network fee (GAS)  
0

Invoke Parameters (JSON)  
[{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"change"}, {"type":16,"value": [{"type":5,"value":"23ba2703c53263e8d6e522dc32203339dc8eee9"}, {"type":16,"value": [{"type":7,"value":"tokens"}, {"type":2,"value":"5"}]}]}]

Caller (AddressBase58)  
APLJBPhRg2XLhtpxEHd6: ▾

No file chosen

Legend: Invoke.

Source: The author, 2019.

Figure 47 - The request of *M3* to change *M1*'s tokens (conclusion).

```
{
  "jsonrpc": "2.0",
  "id": 5,
  "result": {
    "txid": "0xa61652979821d14524733391f5b80aa6b4977d3da1ac85a234b472b80b68052",
    "executions": [
      {
        "trigger": "Application",
        "contract": "0xd5a535f192be1b297c62f17672a7402611bdc2d6",
        "vmstate": "HALT",
        "gas_consumed": "4.554",
        "stack": [
          {
            "type": "ByteArray",
            "value": "5264315750566a417131357a336d57516679736a6f453733685952585152"
          }
        ],
        "notifications": [
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "70726f63657373"
                },
                {
                  "type": "ByteArray",
                  "value": "5264315750566a417131357a336d57516679736a6f453733685952585152"
                },
                {
                  "type": "ByteArray",
                  "value": "546865207265666572656e64756d2070726f636573732068617320737461727465642e"
                }
              ]
            }
          },
          {
            "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
            "state": {
              "type": "Array",
              "value": [
                {
                  "type": "ByteArray",
                  "value": "70726f63657373"
                },
                {
                  "type": "ByteArray",
                  "value": "5264315750566a417131357a336d57516679736a6f453733685952585152"
                },
                {
                  "type": "ByteArray",
                  "value": "5265717565737420746865206368616e6765206f662061206d656d62657227732072656769
→ 7374726174696f6e20646174612e"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

Legend: Output (JSON RPC).

Source: The author, 2019.

Figure 48 - *M1* vote in the referendum about the change request for *M1*.

Invoke Parameters (JSON)

```
[{"type":5,"value":"23ba2703c53263e8d6e522dc32203339dc8eee9"}, {"type":7,"value":"vote"}, {"type":16,"value": [{"type":7,"value":"Rd1WPVjAq15z3mWQfyjoE73hYRXQR"}, {"type":7,"value":true}]}]
```

(a) Invoke.

```
[{"jsonrpc": "2.0",
"id": 5,
"result": {
"txid": "0xff3e993dcf6ef7bbe071c880504a86a40007d723e9125f8d7fa55906bd5418f0",
"executions": [
{
"trigger": "Application",
"contract": "0x204cbbb1f14ab7635c132f0b3af9c6687f7ae8df",
"vmstate": "HALT",
"gas_consumed": "4.033",
"stack": [
{
"type": "Integer",
"value": "1"
}
],
"notifications": [
{
"contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
"state": {
"type": "Array",
"value": [
{
"type": "ByteArray",
"value": "62616c6c6f74"
},
{
"type": "ByteArray",
"value": "5264315750566a417131357a336d57516679736a6f453733685952585152"
},
{
"type": "ByteArray",
"value": "23ba2703c53263e8d6e522dc32203339dc8eee9"
},
{
"type": "ByteArray",
"value": "74727565"
}
]
}
]
}
]
}
}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

Figure 49 -  $M_3$  vote in the referendum about the change request for  $M_1$ .

The screenshot shows a user interface for interacting with a smart contract via neon.js. The top bar is blue with the title "Interacting with a Smart Contract - Invoke". Below it, under "Invoke via neon.js", there are fields for "Contract ScriptHash" (2e4a1cc6ffdc23b5b217d615), "System fee (GAS)" (0), "Network fee (GAS)" (0), and "Invoke Parameters (JSON)" containing a JSON array of objects representing the referendum vote. There are also fields for "Caller (AddressBase58)" (APLJBPhtrRg2XLhtpxEHd6...), an "ABI" section with a file input field, and buttons for "INVOKE", "+ ADD PARAMS", and "Advancing signing".

(a) Invoke.

```
[{"jsonrpc": "2.0",
 "id": 5,
 "result": {
 "txid": "0x5c550b355e4232d444dfcf2ebbfbc8490c8f9f54c3d140d7ef8511577479ca0a",
 "executions": [
 {
 "trigger": "Application",
 "contract": "0x745b1e0d7ab409995756523e5478373ad230350f",
 "vmstate": "HALT",
 "gas_consumed": "4.033",
 "stack": [
 {
 "type": "Integer",
 "value": "1"
 }
 ],
 "notifications": [
 {
 "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
 "state": {
 "type": "Array",
 "value": [
 {
 "type": "ByteArray",
 "value": "62616c6c6f74"
 },
 {
 "type": "ByteArray",
 "value": "5264315750566a417131357a336d57516679736a6f453733685952585152"
 },
 {
 "type": "ByteArray",
 "value": "52eaab8b2aab608902c651912db34de36e7a2b0f"
 },
 {
 "type": "ByteArray",
 "value": "74727565"
 }
 ]
 }
 ]
 }
 ]}]})}]}}
```

(b) Output (JSON RPC).

Source: The author, 2019.

Figure 50 - The result for the  $M3$  change request about  $M1$ .

**Interacting with a Smart Contract - Invoke**

Invoke via neon-js

Contract ScriptHash  2e4a1cc6ffdc23b5b217d615	Invoke Parameters (JSON)  [{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"change result"}, {"type":16,"value":[]}, {"type":7,"value":"Rd1WPVjAq15z3mWQfyjoE73hYRXQR"}]]
System fee (GAS)  0	Network fee (GAS)  0
.....	
Caller (AddressBase58)  APLJBPhRg2XLhtpxEHd6:▼	<input type="button" value="UPLOAD ABI"/> <input type="button" value="Choose File"/> No file chosen
<input type="button" value="(INVOKE)"/> <input type="button" value="+ ADD PARAMS"/>	<input type="checkbox"/> Advancing signing

(a) Invoke.

```
[{"jsonrpc": "2.0",
 "id": 5,
 "result": {
   "txid": "0x7e05056ffd4553716492ae65f2397fa7223ea4ad04a959a551c94e2ef44dfc23",
   "executions": [
     {
       "trigger": "Application",
       "contract": "0x791394325bd8d2333e4982c6547a6203b89aa940",
       "vmstate": "HALT",
       "gas_consumed": "6.522",
       "stack": [
         {
           "type": "Integer",
           "value": "1"
         }
       ],
       "notifications": [
         {
           "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
           "state": {
             "type": "Array",
             "value": [
               {
                 "type": "ByteArray",
                 "value": "70726f63657373"
               },
               {
                 "type": "ByteArray",
                 "value": "5264315750566a417131357a336d57516679736a6f453733685952585152"
               },
               {
                 "type": "ByteArray",
                 "value": "417070726f7665642e"
               }
             ]
           }
         }
       ],
       "contract": "0x2e4a1cc6ffdc23b5b217d615a1ce118ff6b85ff0",
       "state": {
         "type": "Array",
         "value": [
           {
             "type": "ByteArray",
             "value": "6368616e6765"
           },
           {
             "type": "ByteArray",
             "value": "526567697374726174696f6e20646174612e"
           },
           {
             "type": "ByteArray",
             "value": "23ba2703c53263e8d6e522dc32203339dc8eee9"
           }
         ]
       }
     }
   ]
 }
```

(b) Output (JSON RPC).

Source: The author, 2019.

## STEP 6 The trade between members

The last remaining operation to perform a transparent and secure energy exchange agreement. Figure 20 shown in Section 3.4 closes the example and its replication is avoided here. Instead, the next item presents the consult about each member information and how the new values were recorded in the ledger of the test environment.

## STEP 7 The summary operation

The following operations are only available for members, so Figure 51 and Figure 53 have restricted invocation methods. Although the full report option is locked for users, they can still check for any quota and tokens a given *address* may have through the monitoring of the ledger with the APIs available.

Figure 51 - The final values of *M1* (to be continued).

Interacting with a Smart Contract - Invoke

Invoke via neon-js

Contract ScriptHash  
2e4a1cc6ffdc23b5b217d615

System fee (GAS)  
0

Network fee (GAS)  
0

Invoke Parameters (JSON)

```
[{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"summary"}, {"type":16,"value":5}, {"type":5,"value":"23ba2703c53263e8d6e522dc32203339dcdeee9"}, {"type":7,"value":null}]}
```

Caller (AddressBase58)  
APLJBPhRg2XLhtpxEHd6: ▾

Choose File No file chosen

Advancing signing

Legend: Invoke.

Source: The author, 2019.

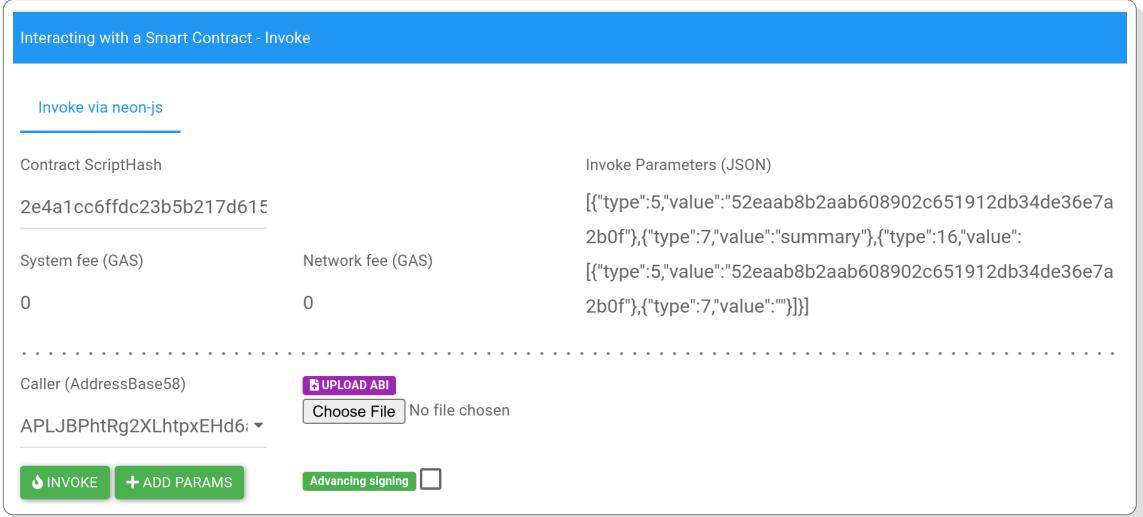
Figure 52 - The final values of  $M1$  (conclusion).

```
[
  {"jsonrpc": "2.0",
   "id": 5,
   "result": {
     "txid": "0x4ef7ed8008a9633df86f4bcecc8d3873794f219ef5033289b61d19c184f16b8b",
     "executions": [
       {
         "trigger": "Application",
         "contract": "0x78a38c49b583f26fd9bc770018b3b03d5774d116",
         "vmstate": "HALT",
         "gas_consumed": "1.504",
         "stack": [
           {
             "type": "Array",
             "value": [
               {
                 "type": "ByteArray",
                 "value": "4d31"
               },
               {
                 "type": "ByteArray",
                 "value": "547261646974696f6e616c20456e657267792053797374656d"
               },
               {
                 "type": "ByteArray",
                 "value": "2f"
               },
               {
                 "type": "ByteArray",
                 "value": "01"
               }
             ]
           ],
           "notifications": []
         }
       }
     ]
   }
]
```

Legend: Output (JSON RPC).

Source: The author, 2019.

Figure 53 - The final values of  $M3$ .



The screenshot shows the "Interacting with a Smart Contract - Invoke" interface. It has a blue header bar. Below it, a section titled "Invoke via neon-js" contains fields for "Contract ScriptHash" (2e4a1cc6ffdc23b5b217d615), "System fee (GAS)" (0), and "Network fee (GAS)" (0). To the right, "Invoke Parameters (JSON)" is shown as an array of objects: [{"type":5,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":"summary"}, {"type":16,"value":"52eaab8b2aab608902c651912db34de36e7a2b0f"}, {"type":7,"value":""}]. Below these fields are "Caller (AddressBase58)" (APLJBPhRg2XLhtpxEHd6:▼) and a file upload section with "UPLOAD ABI" and "Choose File" buttons. A "No file chosen" message is displayed. At the bottom are three buttons: a green "(INVOKE)" button, a green "+ ADD PARAMS" button, and a green "Advancing signing" button with a checkbox.

(a) Invoke.

```
[{"jsonrpc": "2.0",
  "id": 5,
  "result": {
    "txid": "0x97d3fd3db8ceda0546638629575aa89bd71e1545ff04a650bd656c2093261e5c",
    "executions": [
      {
        "trigger": "Application",
        "contract": "0xb075b17c6657eca6c32cbff10dc28dbf5e07f06c",
        "vmstate": "HALT",
        "gas_consumed": "1.504",
        "stack": [
          {
            "type": "Array",
            "value": [
              {
                "type": "ByteArray",
                "value": "4d33"
              },
              {
                "type": "ByteArray",
                "value": "547261646974696f6e616c20456e657267792053797374656d"
              },
              {
                "type": "ByteArray",
                "value": "35"
              },
              {
                "type": "ByteArray",
                "value": "04"
              }
            ]
          }
        ],
        "notifications": []
      }
    ]
  }
}]}
```

(b) Output (JSON RPC).

Source: The author, 2019.

## Workarounds

Unfortunately, some parts of the code only work well when isolated but not when integrated with the whole smart contract. Therefore, the MTEsm version 1.0 (Appendix E) has no problem when compiled and deployed in the blockchain but is restricted to perform some operations. Since the blockchain VM does not return the error message, it is not possible to identify the cause of this behaviour. However, it is probably a limitation of the setup of the test environment because what happens is a stuck of the number of operations that interface with the private space.

So, when some functions are invoked apart, as a unique smart contract, there is no problem. However, when it is part of the MTEsm, the invoke breaks in the middle of the execution of a function. After a lot of hours wasted trying to solve it, the best alternative was be creative. Then, the first letter of each syllable of the word *workaround* results in *war*, which perfectly defines the feeling to deal with the smart contract. Thus, the following functions were added to complement the MTEsm operation:

**admission war** executes the function *RefWar* to properly register the new referendum data in the MTEsm private space since *Ref* was adjusted to create and store the referendum ID only.

**admission result war** executes the function *AdmissionResultWar* to properly register the new member in the MTEsm private space since *AdmissionResult* was adjusted to calculate and evaluate the request outcome only.

**change war** does the same as *admission war*, respecting its peculiarities.

Finally, the implementation of these workarounds are available online in the same repository commented before in Appendix E but now with the tag version 1.1. Therefore, the full MTEsm code continues available for further tests and discussions beyond the present document.